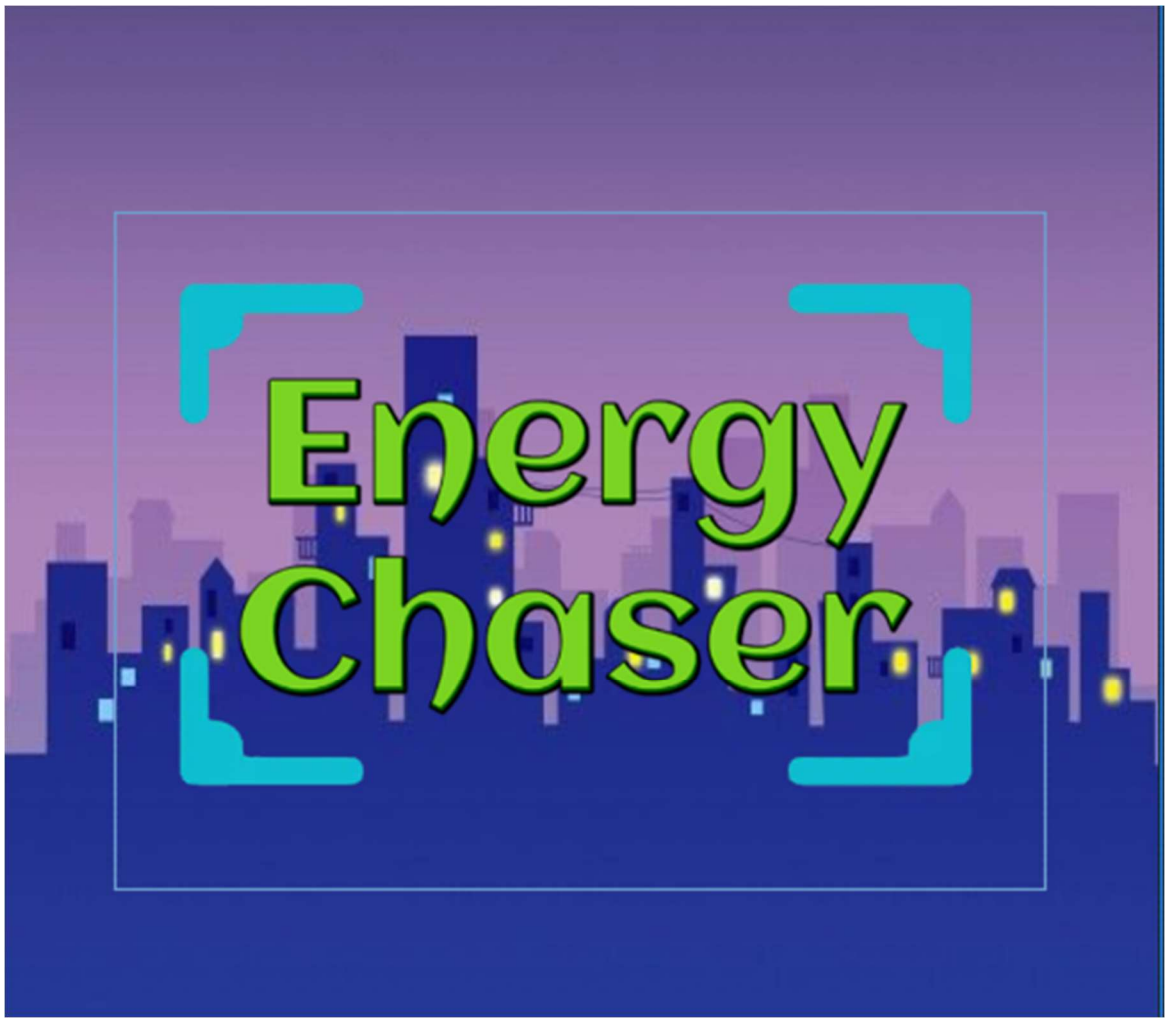
5 Προτεινόμενη Προσέγγιση

5.1 Περίληψη εφαρμογής

Ξεκινώντας να αναφέρω ότι θα ασχοληθώ με την υλοποίηση του βασικού κορμού μιας εφαρμογής Παιχνιδοποίησης για κινητές συσκευές με λειτουργικό Android. Η ιδέα λοιπόν είναι να θυμίζει παιχνίδι, να έχει αρχή και τέλος, να επιβραβεύει τους παίχτες και να κρατάει το ενδιαφέρον τους αμείωτο παράλληλα εκπαιδεύοντας τις γνώσεις τους, πάνω σε ζητήματα ενεργειακής κατανάλωσης. Στο τέλος θα προτείνω επιπλέον λειτουργίες που θα μπορούσαν να μπουν, καθιστώντας την πιο ολοκληρωμένη.



Πιο συγκεκριμένα, το όνομα της εφαρμογής θα είναι το “Energy Chaser” δηλαδή κυνηγός ενέργειας, στην ουσία θα πρόκειται για ένα κυνηγητό, ένα διαδραστικό παιχνίδι πεδίου εκμεταλλευόμενοι την τεχνολογία των φορητών συσκευών όπου ο παίκτης θα αναζητά και θα βρίσκει τις πιο ενεργοβόρες συσκευές σε έναν χώρο και θα τις ταξινομεί κατά βούληση από την πιο ενεργοβόρα στην λιγότερο, μέσα ένα χρονικό διάστημα. Στο τέλος όταν το παιχνίδι έχει την σωστή έκβαση μετά από μια διαδικασία επαλήθευσης, ο παίκτης θα αμείβετε με κάποιο έπαθλο.

5.2 Τεχνολογίες που χρησιμοποιούνται

Για τον εντοπισμό αντικειμένων, θα χρειαστεί να χρησιμοποιήσουμε τεχνολογία μηχανικής μάθησης, μια διαδικασία εκπαίδευσης της μηχανής ώστε να εντοπίζει αντικείμενα, στην περίπτωση μας ηλεκτρικές συσκευές μέσω της κάμερας όταν περνάει μπροστά από τις συσκευές αυτές. Η διαδικασία αυτή υλοποιείτε μέσω εύρεσης του κατάλληλου φωτογραφικού υλικού, ταξινόμησης του σε κλάσεις και σήμανση των συσκευών των οποίων περιλαμβάνονται σε κάθε φωτογραφία, μετά μέσω βιβλιοθηκών της γλώσσας προγραμματισμού Python υλοποιούμε την εκπαίδευση του μοντέλου μας. Η Υλοποίηση Frontend θα γίνει σε γλώσσα σήμανσης XML και το Backend σε γλώσσα προγραμματισμού JAVA. Το τελικό αποτέλεσμα θα είναι η εξαγωγή μιας εφαρμογής Android σε μορφή .apk.

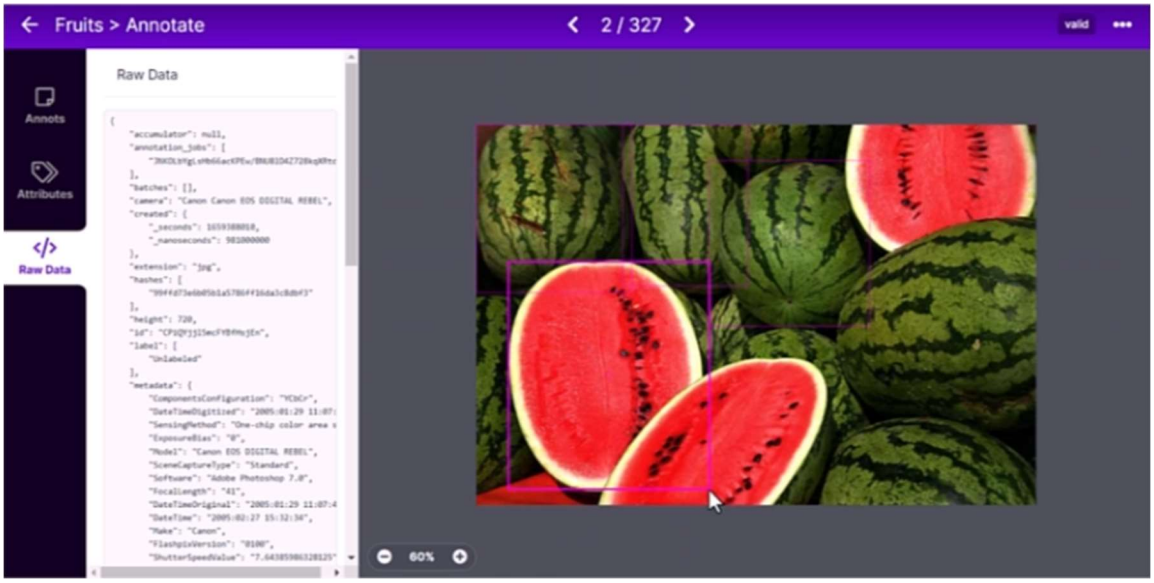
5.3 Σήμανση Συσκευών

Η διαδικασία εντοπισμού αντικειμένων έχει χρήση σε διάφορες εφαρμογές όπως βιντεοπαρακολούθησης, καταμέτρησης πλήθους, ανίχνευσης ανωμαλιών σε γεωργικές βιομηχανίες και φροντίδας υγείας κ.α. .

Η εκπαίδευση του μοντέλου μας θα γίνει μέσω επιβλεπόμενης μάθησης, δηλαδή συλλέγοντας φωτογραφικά δεδομένα μέσω διαδικτύου ή φωτογραφίζοντας τα συνυφασμένα με τα αντικείμενα που θέλουμε να ανιχνεύουμε και σημαδεύοντας τα αντικείμενα αυτά στις φωτογραφίες. Για τις ανάγκες της εφαρμογής, μάζεψα υλικό από

- 1002 Οθόνες.
- 932 Πλυντήρια ρούχων.
- 740 Ψυγεία.
- 413 Φούρνους.
- 360 Φωτιστικά και λάμπες.
- 323 Σταθερούς Υπολογιστές.
- 296 Κλιματιστικά.

Το υλικό είναι διαθέσιμο και σημασμένο στον παρακάτω σύνδεσμο.
https://app.roboflow.com/energy-chaser



Η κάθε συσκευή είναι και μία κλάση, τις φωτογραφίες κάθε κλάσης, μία προς μία τις ελέγχουμε και οριοθετούμε την κάθε συσκευή μέσα σε αυτές με πολύγωνα ή τετράγωνα τα οποία θα βοηθήσουν τον αλγόριθμο μηχανικής μάθησης να εκπαιδεύσει το μοντέλο μας.

Την κάθε κλάση την χωρίσαμε σε 3 κομμάτια.

- 65% Training Set : Το υποσύνολο εκπαίδευσης του αλγορίθμου
- 25% Validation Set : Υποσύνολο για την ρύθμιση των παραμέτρων του αλγορίθμου.
- 10% Testing Set : Υποσύνολο αξιολόγησης του Αλγορίθμου.

Και η εξαγωγή του Dataset μοντέλου έγινε σε Pascal VOC XML δημοφιλή βάση δεδομένων για μοντέλα εκπαίδευσης και ανίχνευσης αντικειμένων με ακρίβεια. Για την σήμανση των συσκευών χρησιμοποίησα το δωρεάν online εργαλείο της roboflow.

5.4 Μηχανική μάθηση(Python)

Η μηχανική μάθηση είναι η παροχή της δυνατότητας σε μία υπολογιστική μηχανή στο να προσαρμόζεται σε άγνωστα περιβάλλοντα και δεδομένα, καθώς επίσης και να ανακαλύπτει πρότυπα και μοντέλα στα δεδομένα αυτά. Λειτουργία η οποία είναι απαραίτητη σε εφαρμογές με υψηλή πολυπλοκότητα πληροφορίας με σκοπό την επίτευξη της βέλτιστης προσαρμογής των παραμέτρων ενός συστήματος για τον σκοπό που έχει επιλεχθεί.

Είναι κλάδος της τεχνητής νοημοσύνης ο οποίος με την χρήση αλγορίθμων και μεθόδων εκπαιδεύει τους υπολογιστές να μαθαίνουν και να καταλαβαίνουν μοτίβα, συμπεριφορές, να εξάγουν και να λαμβάνουν αποφάσεις.

Στην περίπτωση μας θα ακολουθήσουμε την μέθοδο «transfer learning», με την σήμανση αντικειμένων στις εικόνες μέσω του roboflow έχουμε παράξει μια βάση με δεδομένα για κάθε αντικείμενο στην κάθε φωτογραφία κάθε κλάσης.

```xml
<annotation>
    <folder></folder>
    <filename>refrigerators-7-_jpg.rf.9c3493926af3e5f06ac55b21b0504d09.jpg</filename>
    <path>refrigerators-7-_jpg.rf.9c3493926af3e5f06ac55b21b0504d09.jpg</path>
    <source>
        <database>roboflow.ai</database>
    </source>
    <size>
        <width>640</width>
        <height>640</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>refrigerator</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <occluded>0</occluded>
        <bndbox>
            <xmin>39</xmin>
            <xmax>587</xmax>
            <ymin>44</ymin>
            <ymax>592</ymax>
        </bndbox>
    </object>
</annotation>
```

Αυτά τα δεδομένα θα τα χρησιμοποιήσουμε στον αλγόριθμο μηχανικής μάθησης, διαμοιράζοντας έτσι τα βάρη και μεταφέροντας την γνώση βελτιώνοντας έτσι εκθετικά την απόδοση και ακρίβεια της εκπαίδευσης του μοντέλου, εξοικονομώντας παράλληλα χρόνο και πόρους.

Έχοντας λοιπόν στην διάθεση μας όλα τα Datasets θα χρειαστεί μέσω της γλώσσας Python να εκτελέσουμε μια ροή ενεργειών για την εξαγωγή του τελικού εξαγόμενου εκπαιδευμένου μοντέλου σε μορφή Tensorflow lite κατάλληλο για εφαρμογή σε κινητές συσκευές.

Αρχικά θα χρειαστεί να εγκαταστήσουμε 2 απαιτούμενα πακέτα με εργαλεία.

```
!pip install  tflite-model-maker
!pip install  pycocotools
```

Μετά θα χρειαστούμε κάποιες βιβλιοθήκες από τα πακέτα αυτά.

```
[ ]  import numpy as np
     import os
     from tflite_model_maker.config import ExportFormat
     from tflite_model_maker import model_spec
     from tflite_model_maker import object_detector
     import tensorflow as tf
     #assert tf.__version__.startswith('2'
```

«Import numpy as np»

Είναι μια βιβλιοθήκη της Python η οποία παρέχει πλούσιο αριθμό συναρτήσεων και εργαλείων για την δημιουργία επεξεργασία και την ανάλυση αριθμητικών δεδομένων, επιτρέποντας την εκτέλεση πολύπλοκών υπολογισμών με αποδοτικότερη και γρηγορότερη λογική.

«Import os»

Η os είναι μια βιβλιοθήκη με συναρτήσεις διαχείρισης λειτουργικών συστημάτων

Η επόμενες 4 γραμμές εισάγουν κάποιες κλάσεις και συναρτήσεις από τη βιβλιοθήκη tflite_model_maker στη γλώσσα προγραμματισμού Python.

Η κλάση **«ExportFormat»** περιέχει σταθερές που περιγράφουν τα διαθέσιμα φορμάτ εξαγωγής των μοντέλων (π.χ. TensorFlow Lite).

Η συνάρτηση **«model_spec.get()»** επιστρέφει ένα αντικείμενο ModelSpec για ένα συγκεκριμένο προκαθορισμένο μοντέλο. Τα προκαθορισμένα μοντέλα περιλαμβάνουν προ εκπαιδευμένα μοντέλα μηχανικής μάθησης για διάφορες εφαρμογές, όπως η αναγνώριση αντικειμένων, η αναγνώριση κειμένου και άλλα.

Η κλάση **«object_detector»** περιλαμβάνει συναρτήσεις για τη δημιουργία ενός μοντέλου εντοπισμού αντικειμένων. Η διαδικασία αυτή περιλαμβάνει την εκπαίδευση του μοντέλου σε ένα σύνολο δεδομένων εικόνων και την εξαγωγή του μοντέλου σε μια μορφή που μπορεί να χρησιμοποιηθεί σε μια εφαρμογή παραγωγής.

«import tensorflow as tf»

Η εντολή εισάγει τη βιβλιοθήκη TensorFlow στη γλώσσα προγραμματισμού Python δημιουργώντας μια μεταβλητή με όνομα tf που αναφέρεται στη βιβλιοθήκη αυτή.

Η TensorFlow είναι μια ανοιχτού κώδικα βιβλιοθήκη που μπορεί να χρησιμοποιηθεί για τη δημιουργία, την εκπαίδευση και την εκτέλεση μοντέλων μηχανικής μάθησης και βαθιάς μάθησης σε διάφορους τομείς, όπως η αναγνώριση εικόνων.

Στο επόμενο βήμα διευκρινίζουμε την διαδρομή των train validation testing φακέλων με τα annotated δεδομένα και την κάθε κλάση, σημειωτέων όλες οι φωτογραφίες με τα δεδομένα τους πλέον είναι σε έναν φάκελο Energy Chaser χωρισμένες ανά κατηγορία train-valid-test, επίσης είναι σημαντικό η κάθε κλάση να είναι γραμμένη ακριβώς όπως έχει επισημανθεί όταν δημιουργούταν.

```
tr_image_dir= 'C:/Users/kkons/Desktop/Energy Chaser/Energy Chaser/train'
tr_image_annotations= 'C:/Users/kkons/Desktop/Energy Chaser/Energy Chaser/train'
val_image_dir='C:/Users/kkons/Desktop/Energy Chaser/Energy Chaser/valid'
test_image_dir= 'C:/Users/kkons/Desktop/Energy Chaser/Energy Chaser/test'
label_map={1: 'AirCondition', 2:'DESKTOP PC', 3: 'Desktop-PC', 4: 'illumination', 5: 'Monitors',6:'oven' ,7: 'Ovens',8:'refrigerator', 9: 'refrigerators',10:'tvmonitor', 11: 'Washing-Machines'}
print(label_map)

# Load data. Data is loaded as tfrecord and stored in the cache_dir location, for fast future use.
train_ds = object_detector.DataLoader.from_pascal_voc(images_dir=tr_image_dir,
                                                      annotations_dir= tr_image_dir,
                                                      label_map=label_map
                                                     )
# Load validation subset.
val_ds = object_detector.DataLoader.from_pascal_voc(images_dir=val_image_dir,
                                                    annotations_dir= val_image_dir,
                                                    label_map=label_map,
                                                   )
# Load validation subset.
test_ds = object_detector.DataLoader.from_pascal_voc(images_dir=test_image_dir,
                                                     annotations_dir= test_image_dir,
                                                     label_map=label_map,
                                                    )

print("Train dataset contains {} images".format(train_ds.__len__()))
```

Και φορτώνουμε τα υποσύνολα με την σειρά, έχουμε και κάποιες Print για ελέγχους.

Στην επόμενη φάση επιλέγουμε το μοντέλο που θα επανα-εκπαιδεύσουμε το μοντέλο μας, σε αυτή την κατηγορία υπάρχουν τέσσερεις τύποι, όπου αυξάνεται η πολυπλοκότητα από το πιο ελαφρύ «0 – 1 – 2 – 3» στο πιο βαρύ. Εγώ επέλεξα το 1 μιας και το δικό μας μοντέλο έχει υποστεί ήδη επεξεργασία και βρίσκεται σε στάδιο πειραματισμού.

```
[ ]  spec = model_spec.get('efficientdet_lite1')
```

Αφού επιλέξουμε το μοντέλο μας, μπαίνουμε στην διαδικασία της εκπαίδευσης, όπου περνάμε το train Dataset, το μοντέλο και πρέπει να πάρουμε κάποιες αποφάσεις, πόσες φωτογραφίες θα περνάνε για εκπαίδευση του μοντέλου την φορά πχ για να ένα Dataset 1000 φωτογραφιών και ένα batch_size=16 θα γίνουν 1000/16≈ 63 σετ, η επόμενη παράμετρος είναι αν θέλουμε να αναπλαισιώσουμε όλα τα δεδομένα ή ένα κομμάτι τυχαίο από αυτά, εγώ εδώ επέλεξα train_whole_model = True, δηλαδή όλα τα δεδομένα, στην επόμενη παράμετρο φορτώνουμε τα επιβεβαιωμένα μοντέλα και τέλος επιλέγουμε τον αριθμό των επιβεβαιωμένων poy θα περνάει ανά εκπαιδευόμενο σετ. Για παράδειγμα για τα 63 σετ, θα γίνουν 63 x 50 περάσματα. Όλες αυτές τις παραμέτρους μπορούμε να τις προσαρμόσουμε βλέποντας πως ανταποκρίνεται το μοντέλο, σε ακρίβεια, απώλεια κλπ, όσο ανεβάζουμε την ακρίβεια και μειώνουμε την απώλεια πχ μικρότερα σετ με

περισσότερα περάσματα, εννοείτε η διαδικασία γίνεται πιο χρονοβόρα και με μεγαλύτερες απαιτήσεις σε υπολογιστική ισχύ.

```
model = object_detector.create(train_ds, model_spec=spec, batch_size=16, train_whole_model=True, validation_data=val_ds,epochs = 50)
```

Με την επόμενη εντολή εξάγουμε το εκπαιδευμένο πλέον μοντέλο σε συμπιεσμένη μορφή μορφής tflite, εδώ μπορούμε να καθορίσουμε και την διαδρομή του εξαγόμενου μοντέλου.

```
[ ]  model.export(export_dir='.')
```

Σε αυτή την φάση μπορούμε να αξιολογήσουμε το μοντέλο μας με την παρακάτω εντολή.

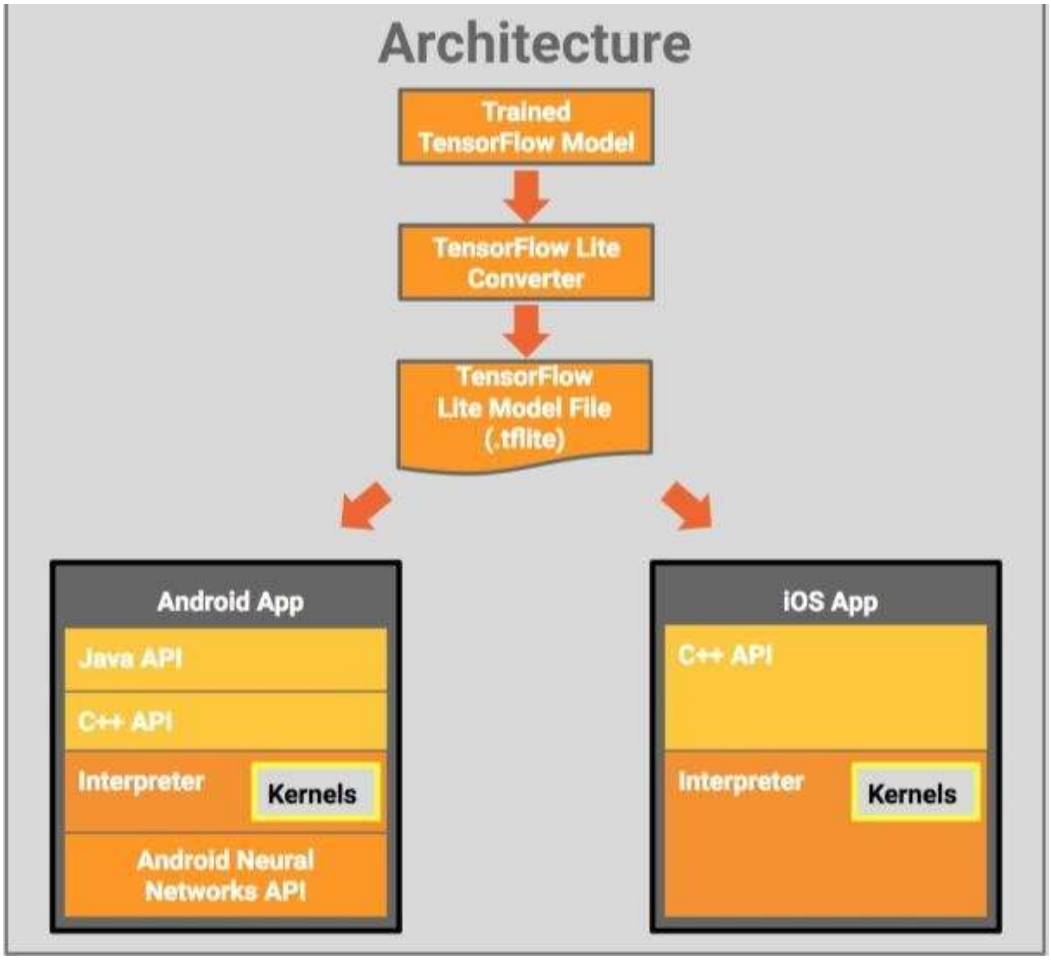
```
model.evaluate_tflite('model.tflite', test_ds)
```

Το αποτέλεσμα από την εκτέλεση της αξιολόγησης είναι μια τιμή μέσης ακρίβειας, αυτή η τιμή κυμαίνεται μεταξύ 0 και 1, όσο πιο κοντά στο 1 πλησιάζει το αποτέλεσμα μας τόσο καλύτερη η πρόβλεψη.

Την διαδικασία της εκπαίδευσης, μπορούμε να την επαναλάβουμε, όσες φορές θέλουμε ή κρίνουμε, μπορούμε να πειραματιστούμε με τις τιμές, ψάχνοντας μια πιο ακριβή πρόβλεψη.

5.5 Tensorflow - Lite

Το TensorFlow Lite είναι μια ελαφριά έκδοση του TensorFlow, σχεδιασμένο ειδικά για χρήση σε κινητές συσκευές, όπως smartphones, tablets, και άλλα embedded συστήματα. Ο στόχος του TensorFlow Lite είναι να παρέχει υψηλές επιδόσεις μηχανικής μάθησης σε συσκευές με περιορισμένους πόρους, χρησιμοποιώντας χαμηλότερο υπολογιστικό φόρτο, μικρότερη μνήμη, χαμηλότερη κατανάλωση ενέργειας και μπορεί να λειτουργεί χωρίς χρήση δικτύου δεδομένων, η τεχνολογία αυτή συχνά αναφέρεται και «at the edge». Αφότου λοιπόν εκπαιδεύσουμε το μοντέλο μας στο TensorFlow το μετατρέπουμε όπως είδαμε στην διαδικασία εκπαίδευσης σε μορφή tensorflow lite(.tflite).



Το μεγαλύτερο προτέρημα του tensorflow lite είναι η **Quantization** μια τεχνολογία συμπίεσης που χρησιμοποιείται στα μοντέλα του. Με την Quantization το tensorflow lite μπορεί με μία διαδικασία μείωσης του αριθμού των bits που απαιτούνται για την αναπαράσταση των δεδομένων μιας μηχανής μάθησης, να μην χάνεται σημαντική πληροφορία. Η Quantization μπορεί να μειώσει σημαντικά το μέγεθος του μοντέλου μηχανικής μάθησης, επιτρέποντας στο TensorFlow Lite να εκτελείται σε συσκευές με περιορισμένη μνήμη.

Η διαδικασία που ακολουθεί είναι να διαιρεί τα βάρη του μοντέλου σε μικρότερα διακριτά εύρη τιμών (π.χ. 8-bit ή 16-bit)1 ή 2 byte αντί για τις συνήθεις 32-bit ή 64-bit(4 ή 8 byte) που χρησιμοποιούνται συνήθως στα μοντέλα μηχανικής μάθησης. Η διαδικασία αυτή πραγματοποιείται με διάφορους τρόπους, συμπεριλαμβανομένης της Post-Training Quantization, κατά την οποία η Quantization εφαρμόζεται μετά την εκπαίδευση του μοντέλου, και η Quantization-aware Training, κατά την οποία η Quantization λαμβάνεται υπόψη κατά τη διάρκεια της εκπαίδευσης.

Μαζί με το μοντέλο tflite θα χρειαστεί να φορτωθεί και ο TensorFlow Lite Interpreter όπου είναι ο πυρήνας του TensorFlow Lite, ο οποίος είναι υπεύθυνος για την εκτέλεση των μοντέλων μηχανικής μάθησης σε κινητές συσκευές και embedded συστήματα. Αναλαμβάνει τη φόρτωση ενός μοντέλου TensorFlow Lite και την εκτέλεση των προβλέψεων στα δεδομένα εισόδου. Ο TensorFlow Lite Interpreter υποστηρίζει μια πληθώρα αρχιτεκτονικών μηχανών, συμπεριλαμβανομένων των ARM, x86, και MIPS, και μπορεί να ενσωματωθεί σε μια ποικιλία εφαρμογών, όπως εφαρμογές αναγνώρισης προσώπων, αναγνώρισης φωνής, αναγνώρισης αντικειμένων και άλλων εφαρμογών μηχανικής μάθησης.

5.6 Frontend – Backend

Για την δημιουργία της εφαρμογής (Frontend - Backend) και την ενσωμάτωση σε αυτή του εκπαιδευμένου μοντέλου αναγνώρισης αντικειμένων, θα χρησιμοποιηθεί το Android Studio ένα ολοκληρωμένο προγραμματιστικό περιβάλλον(IDE) δημιουργίας εφαρμογών σε Android της Google, το οποίο διατίθεται δωρεάν.

Εδώ θα παρουσιασθεί η υλοποίηση του κώδικα παράλληλα σε Java(Backend) και XML(Frontend) καθώς και οι βιβλιοθήκες που χρησιμοποιήθηκαν. Αρχικά θα χρειαστεί να δημιουργηθεί ένα κενό project(Empty Activity) στο Android Studio το οποίο θα χρησιμοποιήσουμε μέχρι τέλους αυτό θα φιλοξενήσει όλη την εφαρμογή, στο όνομα έδωσα το όνομα της εφαρμογής στην περίπτωση μου, επίσης θα επιλεχθεί σαν βασική γλώσσα η JAVA (Το Android Studio υποστηρίζει και την Kotlin).

Με την φόρτωση της εφαρμογής, σκέφτηκα να παίζει ένα μικρό βίντεο animated εισαγωγής σχετικό με την φιλοσοφία της εφαρμογής, ώστε να μυεί τον παίκτη στο Τι θα επακολουθήσει.

Main Activity

Η Main Activity είναι η πρώτη κλάση εδώ φιλοξενώ ένα βίντεο εισαγωγής.

JAVA

```
package com.example.energychaser;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity {
    Timer timer;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

// Εδώ ψάχνει στο XML αρχείο «activity_main» ένα βίντεο βάση id «video_view»,
τύπου VideoView, μόλις το βρει το παίζει αυτόματα.

        VideoView videoView = findViewById(R.id.video_view);
        VideoView.setVideoPath("android.resource://"+getPackageName()+"/"
        +R.raw.entrance);

        videoView.setMediaController(new MediaController((Context) this));
        videoView.start();

// Εδώ θα μας προωθήσει στην επόμενη σελίδα «ButtonsActivity» με καθυστέρηση
14δευτερολέπτων, τα δευτερόλεπτα μπήκαν εσκεμμένα γιατί τόσο διαρκεί το
βίντεο.
        timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                Intent intent = new Intent((Context) MainActivity.this,
               ButtonsActivity.class);
                startActivity(intent);
                finish();
            }
        },14000);
    }
}
```

XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

// Εδώ προσθέτουμε ένα layout το οποίο πατάει στα άκρα και μια κλάση
VideoView (όπου θα φιλοξενεί και το βίντεο)που πατάει πάνω στα άκρα του
layout.

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        tools:ignore="MissingConstraints">
    <VideoView
        android:id="@+id/video_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    </FrameLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

ButtonsActivity.java

Η επόμενη κλάση είναι η «ButtonsActivity» αυτή η κλάση φιλοξενεί την αρχική σελίδα του μενού με τα modes, εγώ εδώ έχω συμπεριλάβει το ατομικό παιχνίδι, παιχνίδι ενός ατόμου όπου ο παίκτης μέσα σε ένα χρονικό διάστημα που επιλέγει, ψάχνει να βρεί συσκευές.

Η επιλογή του Mode γίνεται πατώντας ένα κουμπί, εκτός από το κουμπί σε αυτή την σελίδα έχω βάλει και κάποια Gredits σε popup μορφή για την εφαρμογή.

Επίσης μια imageView κλάση με το logo της εφαρμογής.

JAVA

```
public class ButtonsActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_buttons);
// Εδώ έχουμε ένα κείμενο σε στυλ κουμπιού για να εμφανίσουμε τα Gredits
        TextView clicabletext = (TextView)
this.findViewById(R.id.clicabletext);
        clicabletext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ImageView cancel;
//Ψάχνει το XML αρχείο και πετάει στο κέντρο της σελίδας τα Gredits
                View alertCustomdialog = LayoutInflater.from((Context)
ButtonsActivity.this).inflate(R.layout.credits_popup, null);
//initialize το Gredits box
                AlertDialog.Builder alert = new AlertDialog.Builder((Context)
ButtonsActivity.this);
                alert.setView(alertCustomdialog);
                cancel = (ImageView)
alertCustomdialog.findViewById(R.id.cancel_button);
//Μόλις πατηθεί το εικονίδιο, Ψάχνει την εικόνα βάση ID και κλείνει τα
Gredits
                final AlertDialog dialog = alert.create();
//Εδώ αφαιρεί το background και κάνει να φαίνοται το κουτί των Gredits σαν να
αιωρείται.
                dialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));
//Συνάρτηση που Εμφανίζει το κουτάκι τον Gredits.
                dialog.show();
                cancel.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        dialog.dismiss();
                    }
                });
            }
        });
//Listener Κουμπιου, μόλις πατηθεί μας κάνει προώθηση στην επόμενη σελίδα
"single_player_activity"
        Button button01 = (Button) findViewById(R.id.button1);//single mode
        button01.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openSingleActivity();
            }
        });
    }
    public void openSingleActivity() {
        Intent intent = new Intent((Context) this,
single_player_activity.class);
        startActivity(intent);
    }
}
```

activity_buttons.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:width="3dp"
    android:background="@drawable/first"
    android:color="#00F260"
    tools:context=".ButtonsActivity">

    <Button
        android:id="@+id/button1"
        android:layout_width="312dp"
        android:layout_height="52dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:background="@drawable/buttback"
        android:text="Ατομικό Παιχνίδι"
        app:layout_constraintBottom_toTopOf="@+id/clicabletext"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.828" />

    <TextView
        android:id="@+id/clicabletext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="54dp"
        android:text="Credits"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />

    <ImageView
        android:layout_width="330dp"
        android:layout_height="240dp"
        android:src="@drawable/applogo"
        app:layout_constraintBottom_toTopOf="@+id/button1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.493"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.566"
        tools:ignore="MissingConstraints" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

credits_popup.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:background="@drawable/roundbox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:orientation="vertical"
        android:layout_marginRight="20dp"
        android:layout_marginTop="30dp">

        <ImageView
            android:id="@+id/cancel_button"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:background="@drawable/cancel"
            android:layout_centerHorizontal="true"
            android:layout_marginRight="10dp"
            android:layout_marginTop="5dp"
            android:layout_gravity="right"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20dp"
            android:layout_marginTop="20dp"
            android:layout_marginLeft="20dp"
            android:layout_marginRight="20dp"
            android:orientation="vertical">

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Προιόν Πτυχιακής Εργασίας\n"
                android:textColor="#000"
                android:textStyle="bold"
                />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="Εξοικονόμιση Ενέργειας Στις Εξυπνες
Πόλεις:\nΜηχανισμοί Παιχνιδοποίησης.\nΦοιτητής : Κωνσταντίνος
Κωνσταντινάκος\nΕπιβλέπον Καθηγητής :\nΠαναγιώτης Κόκκινος"
                android:gravity="center"
                android:textColor="#000"
                android:textStyle="bold|italic"
                />

            <Button
                android:layout_width="wrap_content"
```

single_player_activity

Αυτή η κλάση περιλαμβάνει οδηγίες για το ατομικό παιχνίδι σε ένα TextView. Μια TextView που εμφανίζει τα δευτερόλεπτα (Default 20΄΄) όπου σε συνεργασία με μια Seekbar επιλέγουμε δευτερόλεπτα, για την αντίστροφη μέτρηση και τέλος ένα κουμπί αφού έχουν επιλεχθεί τα δευτερόλεπτα να ξεκινήσει το «Ψάξιμο».

```
public class single_player_activity extends AppCompatActivity
{
    SeekBar seekBarsingle;
    TextView textViewsingle;
    private Integer countsec = 20;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_single_player);
        seekBarsingle = (SeekBar)findViewById(R.id.seekBarSingle);
        textViewsingle = (TextView)findViewById(R.id.textViewSingle);

seekBarsingle.setOnSeekBarChangeListener(newSeekBar.OnSeekBarChangeListener()
        {
            @Override
            public void onProgressChanged(SeekBar seekBar, int sec,boolean b)
            {
// Όσο τραβάμε την seekbar αυξάνουμε ή μειώνουμε τον χρόνο παιχνιδιού ανα 5
δευτερόλεπτα
                sec = sec / 5;
                sec = sec * 5;
// Εδω εμφανίζονται τα δευτερόλεπτα στην textViewSingle
                textViewsingle.setText("Δευτερόλεπτα :" + sec);
// Περνάμε τα δευτερόλεπτα σε μια μεταβλητή.
                countsec = sec;
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
        Button camerascan = (Button) findViewById(R.id.camerascan);
        camerascan.setOnClickListener(new View.OnClickListener() {
// Με το πάτημα του κουμπιού μετατρέπουμε τα δευτερόλεπτα σε χιλιοστά του
δευτερολέπτου και στέλνουμε την τιμή με intent.PutExtra στην επόμενη κλάση,
για να αρχίσει η αντιστροφη μέτρηση, παράλληλα προωθείτε και όλη η
δραστηριότητα στην activity_scan_list, όπου αρχίζει και το ψάξιμο.
            @Override
            public void onClick(View v) {
                openSingleActivity(countsec);
            }
        });
    }
    public void openSingleActivity(int value) {
        Integer countime = value;
        if(countime != null || countime != 0 ){countime = value*1000;}
        else{countime=20000;}
        Intent intent = new Intent(single_player_activity.this,
activity_scan_list.class);
        intent.putExtra("MY_INTEGER", countime);
        startActivity(intent);
    }
}
```

Activity_single_player.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".single_player_activity"
    android:orientation="vertical"
    android:background="@drawable/gradientback">
    <TextView
        android:id="@+id/textViewComm"
        android:layout_width="346dp"
        android:layout_height="258dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="24dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="24dp"
        android:fontFamily="serif-monospace"
        android:gravity="center"
        android:text="Στο Ατομικό παιχνίδι ο μόνος αντίπαλος σου είναι ο
χρόνος\n\nΤι κάθεσαι λοιπόν!!!.....\nΕξασκήσου, Μάθε, Ψάξε και βρές ποιές
συκευές καταναλώνουν την περισσότερη ενέργεια.\n\nΜην ξεχάσεις να πείς σε
όλους τι ανακάλυψες!!!"
        android:textSize="18dp"
        android:textStyle="bold" />
    <SeekBar
        android:id="@+id/seekBarSingle"
        android:layout_width="314dp"
        android:layout_height="62dp"
        android:layout_below="@+id/textViewSingle"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="105dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="105dp"
        android:max="120"
        android:progress="20"
        android:scaleX="2"
        android:scaleY="2" />
    <TextView
        android:id="@+id/textViewSingle"
        android:layout_width="180dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textViewComm"
        android:layout_alignStart="@+id/textViewComm"
        android:layout_alignEnd="@+id/textViewComm"
        android:layout_marginStart="60dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="60dp"
        android:fontFamily="serif-monospace"
        android:gravity="center"
        android:text="Δευτερόλεπτα"
        android:textSize="20dp"
        android:textStyle="bold" />
    <Button
        android:id="@+id/camerascan"
```

Activity_scan_list

Σε αυτή την κλάση θα ενσωματωθεί το μοντέλο αναγνώρισης των συσκευών, οι απαραίτητες ερωτήσεις για άρση περιορισμών χρήσης της κάμερας και θα φορτώσουμε τα απαραίτητα dependencies. Αυτή η κλάση είναι που αρχίζει το παιχνίδι, λόγο του ότι αυτή η κλάση περιέχει πολύ κώδικα, εδώ θα συνοψίσω τις κυριότερες συναρτήσεις που χρειάζονται, πάραυτα στην έκθεση θα συμπεριληφθεί όλος ο κώδικας της εφαρμογής με σχόλια.

Στο frontend αυτή η κλάση περιέχει μια «progress bar» όπου με γραφικό τρόπο θα απεικονίζει τον χρόνο που περνάει, ένα πλαίσιο «Fragment» όπου θα έχουμε εικόνα μέσω κάμερας και ένα κουμπί τύπου «ImageView» όπου όταν αναγνωρίζεται κάποιο αντικείμενο μέσω κάμερας θα μπορούμε να το πατάμε και να καταχωρούμε το αντικείμενο που εντοπίσαμε σε μια λίστα με πέντε(5) μέγιστο αριθμό καταχωρήσεις, με σκοπό να ταξινομηθούν στην επόμενη κλάση.

Scan_activity_list

```
<FrameLayout
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"

tools:context="org.tensorflow.lite.examples.classification.CameraActivity">
</FrameLayout>
<ImageView
    android:id="@+id/addbutton"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_marginBottom="116dp"
    android:onClick="myClick"
    android:shape="rectangle"
    android:src="@drawable/addbut"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent" />
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="155dp"
    android:layout_height="157dp"
    android:indeterminateOnly="false"
    android:progressDrawable="@drawable/progress_bar"
    app:layout_constraintBottom_toBottomOf="@+id/addbutton"
    app:layout_constraintEnd_toEndOf="@+id/addbutton"
    app:layout_constraintStart_toStartOf="@+id/addbutton"
    app:layout_constraintTop_toTopOf="@+id/addbutton" />
```

Στο Backend αρχικά θα χρειαστεί να προσθέσουμε τους περιορισμούς για χρήση της κάμερας. Αυτό θα γίνει στο αρχείο «Android Manifest» όπου θα προσθέσουμε την εξής γραμμή κώδικα.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Επίσης θα χρειαστεί να προσθέσουμε και μια ερώτηση στον κορμό της κλάσης μας, διότι από την Android Marshmallow έκδοση και μετά είναι μια υποχρεωτική διαδικασία.

```
//TODO show live camera footage
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
    if (checkSelfPermission(android.Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_DENIED){
        String[] permission = {Manifest.permission.CAMERA};
        requestPermissions(permission, PERMISSION_CODE);
    }
    else {
        setFragment();
    }
}
```

Εδώ στην περίπτωση που έχει γίνει η άρση των περιορισμών, προχωράμε στην λειτουργεία της αναγνώρισης συσκευών, συγκεκριμένα η μέθοδος setFragment() η οποία λαμβάνει χώρα σε ένα container στο scan_activity_list.xml αρχείο μας ,δημιουργώντας ένα αντικείμενο fragment μέσα στο «CameraConnectionFragment.java» και στο «camera_fragment.xml».

```
                this,
                R.layout.camera_fragment,
                new Size(640, 480));
camera2Fragment.setCamera(cameraId);
fragment = camera2Fragment;
getFragmentManager().beginTransaction().replace(R.id.container,
fragment).commit();
```

Αναλαμβάνοντας έτσι τον έλεγχο τις κάμερας σε ζωντανή λήψη περνώντας τις εικόνες frame by frame στο μοντέλο μας για αναγνώριση και εξάγοντας τα πλαίσια στην εικόνα μας ζωντανά. Για να γίνει λοιπόν αυτό θα χρειαστεί να περαστεί κάθε frame εικόνας σε μορφή bitmap.

Αυτό γίνεται δημιουργώντας ένα CameraConnectionFragment και περνώντας διάφορες παραμέτρους σε αυτό.

```
Fragment fragment;
CameraConnectionFragment camera2Fragment =
        CameraConnectionFragment.newInstance(
```

Στο «CameraConnectionFragment.java» θα γίνει η παραμετροποίηση των εξής παραμέτρων.

```
public static CameraConnectionFragment newInstance(
//Τι επιστρέφει.
        final ConnectionCallback callback,
//Interface (Θα αναλύσουμε στην συνέχεια)
        final OnImageAvailableListener imageListener,
// Το Layout που είναι συνδεδεμένο με το
CameraConnectionFragment(camera_fragment.xml)
        final int layout,
//Το μέγεθος της εικόνας που απεικονίζεται μέσω κάμερας(640x480)
        final Size inputSize) {
    return new CameraConnectionFragment(callback, imageListener, layout,
inputSize);
}
```

Για κάθε frame που θα δημιουργείτε θα καλούμε την μέθοδο «OnImageAvailable()» στην activity_scan_list.java η οποία σχετίζεται με το interface που κάνουμε implement στην βασική μας κλάση.

```
public class activity_scan_list extends AppCompatActivity implements
ImageReader.OnImageAvailableListener{
```

Η μέθοδος δέχεται ως είσοδο το κάθε frame μέσω του ImageReader reader

```
public void onImageAvailable(ImageReader reader) {
```

Το οποίο μας επιστρέφει το κάθε frame σε μορφή εικόνας.

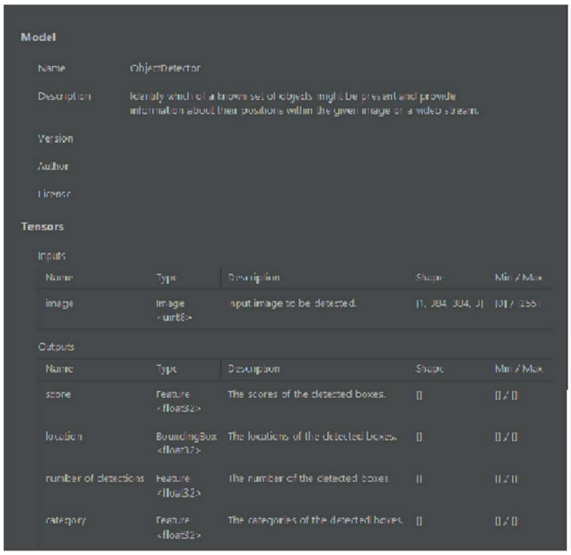
```
final Image image = reader.acquireLatestImage();
```

Και καλεί την processImage για να μετατρέψει την εικόνα σε bitmap μορφή, την μορφή που χρησιμοποιούμε στο μοντέλο αναγνώρισης μας.

```
processImage();
```

Για να μπορεί να γίνει φυσικά η αναγνώριση των συσκευών θα χρειαστεί να φορτωθεί το μοντέλο μας, για να γίνει αυτό θα πρέπει να φτιάξουμε στην εφαρμογή μας έναν νέο φάκελο με assets. Αυτό θα γίνει ως εξής app →New→Folder → Assets Folder

Στην συνέχεια κάνουμε copy – paste το μοντέλο μας με κατάληξη .tflite το οποίο έχουμε εξάγει και θα χρειαστεί να δημιουργήσουμε άλλο ένα αρχείο .txt το οποίο θα περιέχει τις κλάσεις των συσκευών με την σειρά που τις αναφέραμε κατά την εκπαίδευση του μοντέλου στην Python.



EnergyChaser.tflite

```
1  AirCondition
2  DESKTOP PC
3  Desktop-PC
4  illumination
5  Monitors
6  oven
7  Ovens
8  refrigerator
9  refrigerators
10 tvmonitor
11 Washing-Machines
```

lablemap.txt

Στην συνέχεια θα χρειαστεί να φορτώσουμε την βιβλιοθήκη tensorflow-lite στην εφαρμογή μας, για να γίνει αυτό, θα πάμε στο αρχείο build.gradle και θα προσθέσουμε την εξής γραμμή.

```
implementation 'org.tensorflow:tensorflow-lite:+'
```

Και θα ξανά συγχρονίσουμε το προτζεκτ μας κάνοντας Synch Project with Gradle Files.

Αυτό μας εισάγει ένα Interface(Detector) και ένα API(TFLiteObjectDetectionAPIModel) το οποίο μας επιτρέπει να εισάγουμε εικόνα στο μοντέλο μας και να πάρουμε το εξαγόμενο αποτέλεσμα μέσω του αλγόριθμου που εκπαιδεύσαμε.

Οι δύο βασικότερες μέθοδοι που περιέχει το API μας είναι πρώτον η Static create Detector, αυτή η κλάση δημιουργεί ένα tflite object

```
public static Detector create(
    final Context context,      (τα περιεχόμενα του)
    final String modelFilename,(energychaser.tflite)
    final String labelFilename,(labelmap.txt)
    final int inputSize,        (Το size της εικόνας, περιέχεται στο .tflite
384x384)
    final boolean isQuantized), (Αν το μοντέλο μας έχει συμπίεση Quantize
uint8 επίσης στο .tflite)
    throws IOException {
  return new TFLiteObjectDetectionAPIModel(context, modelFilename);
}
```

Και η recognizeImage όπου δέχεται την εικόνα σε μορφή bitmap και επιστρέφει το αποτέλεσμα από την λίστα με τα περιεχόμενα της κλάσης Recognition() που δηλώνεται στο Interface «Detector», όπου περιέχει τα id, title,confidence,location των αντικειμένων που εντοπίζονται.

```
@Override
public List<Recognition> recognizeImage(final Bitmap bitmap) {
  Trace.beginSection("recognizeImage");
  List<Detection> results =
objectDetector.detect(TensorImage.fromBitmap(bitmap));
  final ArrayList<Recognition> recognitions = new ArrayList<>();
  int cnt = 0;
  for (Detection detection : results) {
    recognitions.add(
        new Recognition(
            "" + cnt++,
            detection.getCategories().get(0).getLabel(),
            detection.getCategories().get(0).getScore(),
            detection.getBoundingBox()));
  }
  Trace.endSection(); // "recognizeImage"
  return recognitions;
}
```

Στην βασική μας κλάση τώρα θα χρειαστεί να κάνουμε κάποιες δηλώσεις.

```
Handler handler;
private Matrix frameToCropTransform;
private int sensorOrientation;
private Matrix cropToFrameTransform;

//Το μέγεθος εικόνας
private static final int TF_OD_API_INPUT_SIZE = 384;

// MINIMUM CONFIDENCE εδώ θέλουμε να μας εμφανίζει της συσκευές από κάποιο
confidence και επάνω, όσο πιο μεγάλο νούμερο τόσο πιο σίγουρο είναι το
αποτέλεσμα, όσο πιο κάτω, μπορεί να έχουμε συσκευές με μεγαλύτερο ποσοστό
λάθος αναγνώρισης.
private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.6f;
// Κάποιες μεταβλητές που θα χρειαστούμε
private static final boolean MAINTAIN_ASPECT = false;
private static final float TEXT_SIZE_DIP = 10;

//Εδώ δηλώνουμε τον Detector και κάνουμε import τις κλάσεις που εισαγάγαμε με
//την βιβλιοθήκη του tensorflow - lite
//import org.tensorflow.lite.examples.detection.tflite.Detector;
//import org.tensorflow.lite.examples.detection.tflite.
//TFLiteObjectDetectionAPIModel;

// αρχικοποιούμε τον Detector από πού θα τραβάει τις πληροφορίες για τα
αντικείμενα που σκανάρει, φορτώνοντας έτσι το μοντέλο μας στην εφαρμογή.

private Detector detector;
```

Και να αρχικοποιήσουμε τον Detector ώστε να φορτώσει τα αρχεία με τις πληροφορίες και τις κλάσεις του μοντέλου.

```
detector = TFLiteObjectDetectionAPIModel.create(
                (Context) this,
                "EnergyChaserl.tflite",
                "labelmap.txt",
                TF_OD_API_INPUT_SIZE,
                true);
```

Στην συνέχεια έχει δημιουργηθεί μια μέθοδο processImage η οποία όπως είδαμε την καλούμε μέσα στην «OnImageAvailable()» όπου μετατρέπει την εισερχόμενη εικόνα σε bitmap μορφή και επιστρέφει μια λίστα με τις αναγνωρίσεις που έχουν γίνει μέσω κάμερας.

Στο συγκεκριμένο σημείο να αναφέρω σε αυτή την μέθοδο ζωγραφίζουμε και τα τετράγωνα στην κάθε συσκευή και τις πληροφορίες που φαίνονται, όνομα συσκευής και το confidence score. Και τα δύο αυτά απεικονίζονται μέσω του camera_fragment.xml η εικόνα στο texture και τα τετράγωνα στο tracking overlay δίνοντας την ψευδαίσθηση ότι είναι ένα.

```
rgbFrameBitmap.setPixels(rgbBytes, 0, previewWidth, 0, 0, previewWidth,
previewHeight);
final Canvas canvas = new Canvas(croppedBitmap);
canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
```

Περνάμε τις αναγνωρίσεις στον Detector.

```
List<Detector.Recognition> results = detector.recognizeImage(rgbFrameBitmap);
```

Και όσα έχουν > 60% Confidence περνάνε στην απεικόνιση της οθόνης του κινητού στο textures

```
float minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
final List<Detector.Recognition> mappedRecognitions =
        new ArrayList<>();
for (final Detector.Recognition result : results) {
    if (result.getConfidence() >= minimumConfidence) {
        mappedRecognitions.add(result);
```

Και εν συνεχεία…

Σε αυτό το σημείο μπαίνει και ένας onClick listener για το κουμπί μας, όπου όταν αναγνωρίζουμε μια συσκευή, με το πάτημα παίρνουμε τον τίτλο της συσκευής και τον περνάμε σε μια λίστα, μεταφράζουμε και συμπυκνώνουμε τις κλάσεις των συσκευών και τέλος κάνουμε έναν έλεγχο για διπλότυπα.

Παράλληλα έχουμε έναν μετρητή που καταγράφει τα έγκυρα εισερχόμενα μέχρι να συμπληρωθούν πέντε στο σύνολο και τέλος έχουμε μια μέθοδο «sendeverything()» που τα στέλνει όλα στην επόμενη κλάση για επεξεργασία.

```
public void processImage(){
    imageConverter.run();;
////////////////////////////////////////////////////////////////////////////////
Αναλύθηκαν πιο πάνω
////////////////////////////////////////////////////////////////////////////////
                    if (objectCount < 5) {
//Εδώ με το πάτημα του κουμπιού περνάμε τις συσκευές που αναγνωρίστηκαν στην
λίστα μας, περνόντας όμως αρχικά από κάποιους ελέγχους
                        ImageView imgButton=findViewById(R.id.addbutton);
imgButton.setOnClickListener(newView.OnClickListener(){
                            public void onClick(View v) {
//Αποθηκεύεται ο Τίτλος σε μια string μεταβλητή
                                String Device = result.getTitle();
////////////////////////////////////////////////////////////////////////////////
Σε αυτό το σημείο γίνεται μετάφραση των κλάσεων του μοντέλου μας, μιας και
απευθυνόμαστε σε Ελληνικό κοινό.
////////////////////////////////////////////////////////////////////////////////
Και εδώ ελέγχουμε για διπλότυπα
                                boolean found = false;
                                for (String item : objects ){
                                        if (item.equals(Device)){
                                            found = true;
                                            break;
                                        }}
                                if(found){
                                    Toast.makeText(activity_scan_list.this,
"Έχετε προσθέσει ξανα τη συσκευή " + Device, Toast.LENGTH_SHORT).show();
                                }
                                else{
                                    objects.add(Device
                                    objectCount++;
                                    objectCoCount--;
                                    Toast.makeText(activity_scan_list.this,
"Προστέθηκε : " + Device, Toast.LENGTH_SHORT).show();
                                    Toast.makeText(activity_scan_list.this,
"Μένουν Άλλα : " + objectCoCount, Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                    }
                    else {
                        Toast.makeText(activity_scan_list.this,
"Ολοκληρώθηκαν Οι Προσθήκες!", Toast.LENGTH_SHORT).show();
                        sendeverything();
                    }
                }
            }
            tracker.trackResults(mappedRecognitions, 10);
            trackingOverlay.postInvalidate();
            postInferenceCallback.run();
        }
    });
}
```

Στην αρχή και στο τέλος κάθε επανάληψης καλούμε τρεις μεθόδους…

```
tracker.trackResults(mappedRecognitions, 10);
trackingOverlay.postInvalidate();
postInferenceCallback.run();
```

Η πρώτη είναι για την απεικόνιση των τετραγώνων στην οθόνη, αυτό γίνεται όπως είπαμε frame by frame, αυτή την λειτουργεία την αναλαμβάνει στην «OnImageAvailable()» η Boolean μεταβλητή «isProcessingFrame» .

```
//Αν η μεταβλητή είναι True, σηματοδοτεί ότι αυτό το Frame έχει επεξεργαστεί,
έτσι επιστρέφουμε πάνω και ελέγχουμε το επόμενο. Αν στο επόμενο είναι False…
if (isProcessingFrame) {
    image.close();
    return;
}
//Ερχόμαστε εδώ και αλλάζουμε την μεταβλητή σε True ώστε μέχρι να
ολοκληρώσουμε την επεξεργασία αυτού του Frame να μην περάσει κάποιο άλλο.
isProcessingFrame = true;
final Image.Plane[] planes = image.getPlanes();
fillBytes(planes, yuvBytes);
yRowStride = planes[0].getRowStride();
final int uvRowStride = planes[1].getRowStride();
final int uvPixelStride = planes[1].getPixelStride();
```

Η μεταβλητή αφού γίνει η επεξεργασία γίνεται «False» σε αυτό το σημείο…λίγο πιο κάτω.

```
postInferenceCallback =
        new Runnable() {
            @Override
            public void run() {
                image.close();
                isProcessingFrame = false;
            }
        };
```

Και καλούμε την Τρίτη μέθοδος postInferenceCallback.run(); Όπου βάζουμε το επόμενο Frame μέσω Detector.

Η δεύτερη είναι μια μέθοδος της Java όπου προκαλεί μια ακύρωση σε έναν επόμενο κύκλο μέσω του βρόχου συμβάντος, μέχρι να ολοκληρωθεί ο προηγούμενος.

Για όλα τα παραπάνω υπάρχουν τα ανάλογα μηνύματα, παράλληλα με την είσοδο στην κλάση έχει αρχίσει αυτόματα και μετράει ο χρόνος αντίστροφα μέσω της μεθόδου «Countime()».

```
public void countime(){
// Μέθοδο χρονομέτρησης
    Timer = new CountDownTimer(passedSecs, 1000) {
        @Override
        public void onTick(long millisUntilFinished) {
            CurrentProgress = (int) ((passedSecs - millisUntilFinished) * 100
/ passedSecs);
            progressBar.setProgress(CurrentProgress);
            progressBar.setMax(100);
//Αυτή η μεταβλητή μεταφέρεται στην επόμενη κλάση, για να συνεχίσει να
μετράει ο χρόνος από το σημείο που αφήσαμε σε αυτή την κλάση
            remain = millisUntilFinished;
        }
        @Override
        public void onFinish() {
            progressBar.setProgress(100);
            isTimerRunning = false;

                Timer.cancel();
//Σε περίπτωση που τελειώσει ο χρόνος, χωρίς να έχουμε ολοκληρώσει τις
προσθήκες, καθαρίζουμε την λίστα με τα Objects
                objects.clear();
// Μήνυμα τέλους χρόνου
                Toast.makeText(activity_scan_list.this, "Time's up!",
Toast.LENGTH_SHORT).show();
// Και μετά από 1 δευτερόλεπτο γίνεται redirect στην προηγούμενη κλάση
                Runnable t = new Runnable() {
                    @Override
                    public void run() {
                        startActivity(new Intent(activity_scan_list.this,
ButtonsActivity.class));
                    }
                };
                Handler h = new Handler();
                h.postDelayed(t, 1000);
        }
    };
    Timer.start();
    isTimerRunning = true;
}
// Αυτή η μέθοδος είναι για να αρχίζει ο χρόνος αυτόματα
@Override
protected void onResume() {
    super.onResume();
    // resume the timer if it is running
    if (isTimerRunning) {
        Timer.start();
    }
```

Η «sendeverything()» .

```
public void sendeverything(){
//Η μέθοδο αυτή χρησιμοποιείτε εφόσον έχουν ολοκληρωθεί όλες οι προσθήκες
επιτυχώς.

//Σετάρει την progressbar στο 100%
    progressBar.setProgress(100);
//Σταματάει τον χρόνο για την στιγμή
    isTimerRunning = false;
//Μας προωθεί στην επόμενη κλάση και στέλνει υπολοιπόμενο χρόνο και την λίστα
με τις αναγνωρίσεις.
    Intent intent = new Intent(this, CheckList.class);
    intent.putStringArrayListExtra("objectList", objects);
    intent.putExtra("remainingTime", remain);
    startActivity(intent);
}
```

Η «getScreenOrientation()».

```
protected int getScreenOrientation() {
Μέθοδος για την περιστροφή των frames(Κουτάκια αναγνώρισης) σε πορτραίτο ή σε
κάθετη απεικόνιση.
    switch (getWindowManager().getDefaultDisplay().getRotation()) {
        case Surface.ROTATION_270:
            return 270;
        case Surface.ROTATION_180:
            return 180;
        case Surface.ROTATION_90:
            return 90;
        default:
            return 0;
    }
}
```

Η Απαραίτητη «OnDestroy()» για την απελευθέρωση της RAM.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    detector.close();
}
```

Και αισίως μπαίνουμε στην τελευταία κλάση.

CheckList.java

Σε αυτή την κλάση ο παίκτης θα χρειαστεί να ταξινομήσει τις συσκευές που συνέλεξε από την πιο ενεργοβόρα στην λιγότερο ενεργοβόρα ενώ συνεχίζει ο χρόνος να τρέχει. Για αυτό το λόγο θα χρειαστεί να εμφανίσουμε την λίστα με τους τίτλους των συσκευών σε μια RecyclerView.

Η "RecyclerView" αναφέρεται σε έναν προηγμένο τύπο λίστας που χρησιμοποιείται για την εμφάνιση μιας λίστας στοιχείων σε ένα UI στοιχείο. Η RecyclerView χρησιμοποιείται για την αντικατάσταση του παλαιότερου ListView στην ανάπτυξη εφαρμογών Android. Προσφέρει πιο ευέλικτες δυνατότητες και απόδοση.

Μαζί με την RecyclerView πάνε πακέτο ένας ViewHolder και έναν Adapter ο οποίος είναι υπεύθυνος για την παροχή δεδομένων στο RecyclerView και τη δημιουργία των ViewHolder. Ο Adapter λαμβάνει τα δεδομένα από μια πηγή ViewHolder ο οποίος αναπαριστά ένα στοιχείο του RecyclerView. Αντιστοιχεί στη γραφική αναπαράσταση του στοιχείου στη λίστα. Ο σκοπός του ViewHolder είναι να αποθηκεύει αναφορές στα στοιχεία της διεπαφής χρήστη και να παρέχει πρόσβαση σε αυτά για να ενημερωθούν με δεδομένα από τον Adapter.

Με την δημιουργία της RecyclerView δημιουργούνται αυτόματα και κάποιες μέθοδοι.

Ένας CustomAdapter.

```
public CustomAdapter(ArrayList<String> items, RecyclerView recyclerView) {
    super(items, recyclerView, ItemTouchHelper.UP | ItemTouchHelper.DOWN, 0);
}
```

onBindViewHolder() η οποία είναι υπεύθυνη για την προβολή της λίστας στην RecyclerView, και τη θέση (position) κάθε στοιχείου στη λίστα δεδομένων.

```
@Override
public void onBindViewHolder(CustomViewHolder holder, int position) {
    holder.textView.setText(DetectedList.get(position));
}
```

Και η onMove()

```
public void onMove(CustomViewHolder viewHolder, CustomViewHolder target) {
    //Toast.makeText(getBaseContext(), "Item " +
viewHolder.getBindingAdapterPosition() + " moved",
Toast.LENGTH_SHORT).show();
    String s = DetectedList.remove(viewHolder.getBindingAdapterPosition());
    DetectedList.add(target.getBindingAdapterPosition(), s);

//Κάθε φορα που μετακινούμε ένα στοιχείο τις λίστας στον RecyclerView θα
πρέπει να μετακινείτε και στην ίδια την λίστα, αυτό γίνεται εδώ.
    int fromPosition = viewHolder.getBindingAdapterPosition();
    int toPosition = target.getBindingAdapterPosition();
//Απομάκρυνση από την θέση αυτή διαγράφοντας το, αφού πρώτα το έχουμε περάσει
σε μια String μεταβλητή.
    String item = DetectedList.remove(fromPosition);
//Και τοποθετόντας το στην νέα θέση.
    if (toPosition > fromPosition) {
        toPosition--;
    }
    DetectedList.add(toPosition + 1, item);
// Και ενημερώνουμε για την αλλαγή.
    notifyItemMoved(fromPosition, toPosition + 1);
//Αντιγράφη τις λίστας σε μια global λίστα.
    reordered = new ArrayList<>();
    reordered.addAll(DetectedList);
}
```

Ένας viewHolder ο οποίος θα γίνει implement στην recyclerView και μας επιτρέπει με κάθε παρατεταμένο κλικ πάνω στα στοιχεία της λίστας μας επιτρέπει να μετακινούμε τα στοιχεία ενημερώνοντας παράλληλα για αλλαγές στις θέσεις και backend αλλά και front σε ένα στοιχείο textview στο XML αρχείο μας.

```
public class CustomViewHolder extends RecyclerView.ViewHolder {
    TextView textView;
    public CustomViewHolder(View itemView) {
        super(itemView);
        textView = (TextView) itemView.findViewById(R.id.textview);
        itemView.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View view) {
                    startDrag(CustomViewHolder.this, getLayoutPosition());
                return true;
            }
        });
    }
}
```

Εφόσον έχουμε ολοκληρώσει τις μετακινήσεις μέσα στο χρόνο που αναλαμβάνει η μέθοδος «countime()» που είδαμε και στην προηγούμενη κλάση, πατώντας το κουμπί της ολοκλήρωσης.

```
Button finalizeButton = (Button) findViewById(R.id.finalize_button);
finalizeButton.setOnClickListener(v -> Finilize(reordered));
```

Καλούμε και την μέθοδο Finalize(reorderd) η οποία παίρνει ως είσοδο μια λίστα(την ταξινομημένη).

```
private void Finilize(ArrayList<String> list) {

//Σταματάει τον Χρόνο
    isTimerRunning = false;
//Ελέγχει μέσω της παρακάτω μεθόδου αν η ταξινομημένη λίστα είναι σε σωστή
σειρά.
    boolean areInOrder = areItemsInOrder(savedList, list);
//Εάν είναι βγάζει μήνυμα επιβράβευσης και κάνει ορατό μια κινούμενη εικόνα
gif.
    if (areInOrder) {
        Toast.makeText(CheckList.this, "Μπράβο!!!.Τα Κατάφερες Μια Χαρά!!!",
Toast.LENGTH_SHORT).show();
        imageView.setVisibility(View.VISIBLE);
        backtostart();
    }
//Διαφορετικά μήνυμα λάθους.
    else{
        Toast.makeText(CheckList.this, "Την Επόμενη Φορά Θα Πάει Σίγουρα
Καλύτερα!!!", Toast.LENGTH_SHORT).show();
        backtostart();
    }
//Και στις δύο περιπτώσεις έχουμε την backtostart η οποία μετά από ένα
χρονικό διάστημα μας επιστρέφει στο αρχικό μενού.
}
```

Για να γίνει ο έλεγχος της ταξινομημένης λίστας χρειαζόμαστε μια λίστα με την σωστή σειρά, η οποία είναι global και την έχουμε δηλώσει στην onCreate().

```
savedList = new ArrayList<>();
//List of objects to compare
savedList.add("ΦΟΥΡΝΟΣ");
savedList.add("ΨΥΓΕΙΟ");
savedList.add("ΠΛΥΝΤΗΡΙΟ");
savedList.add("ΚΛΙΜΑΤΙΣΤΙΚΟ");
savedList.add("ΥΠΟΛΟΓΙΣΤΗΣ");
savedList.add("ΟΘΟΝΗ");
savedList.add("ΦΩΤΙΣΜΟΣ");
```

Η μέθοδος ελέγχου ελέγχει τις δύο λίστες και επιστρέφει True αν είναι όλα σωστά False αν είναι λάθος.

```
public static boolean areItemsInOrder(ArrayList<String> list1,
ArrayList<String> list2) {
    int i = 0;
    int j = 0;
    while (i < list1.size() && j < list2.size()) {
        if (list1.get(i).equals(list2.get(j))) {
            j++;
        }
        i++;
    }
    return j == list2.size();
}
```

Στην onCreate() έχουμε την εικόνα ώς αόρατη μέχρι να αλλάξει σε ορατή μέσω της finalize()

```
imageView = findViewById(R.id.awardgif);
imageView.setVisibility(View.INVISIBLE);
```

Και η Backtostart()

```
public void backtostart(){
    Runnable t = () -> {
        reordered.clear();
        DetectedList.clear();
        Timer.cancel();
        startActivity(new Intent(CheckList.this, ButtonsActivity.class));
    };
    Handler h = new Handler();
Μετά από 8 δευτερόλεπτα, επιστροφή στην αρχή.
    h.postDelayed(t, 8000);
}
```

Στο Frontend έχουμε.

- TextView με πληροφορίες
- ProgressBar που μετράει τον χρόνο
- Button για την ολοκλήρωση της ταξινόμησης.
- GifImageView την εικόνα που εμφανίζεται μετα την επιτυχή έκβαση της διαδικασίας
- recylerView που φιλοξενεί την λίστα προς ταξινόμηση.