Diseases Detection in Citrus Fruits Using Convolutional Neural Networks

Fernando López Laso
TFG – Enginyeria Informàtica

Control

Importació ↔ Exportació

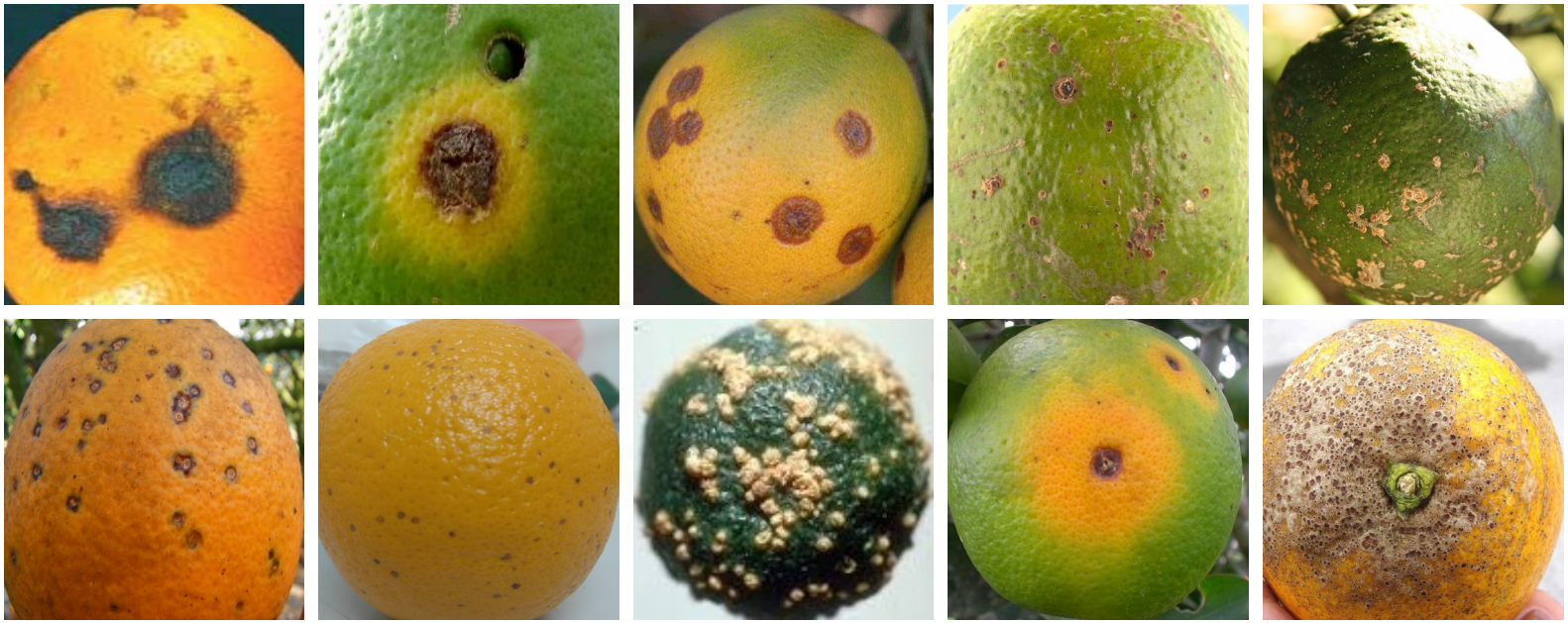
Gran varietat de productes. Molts símptomes per producte.

Cítrics

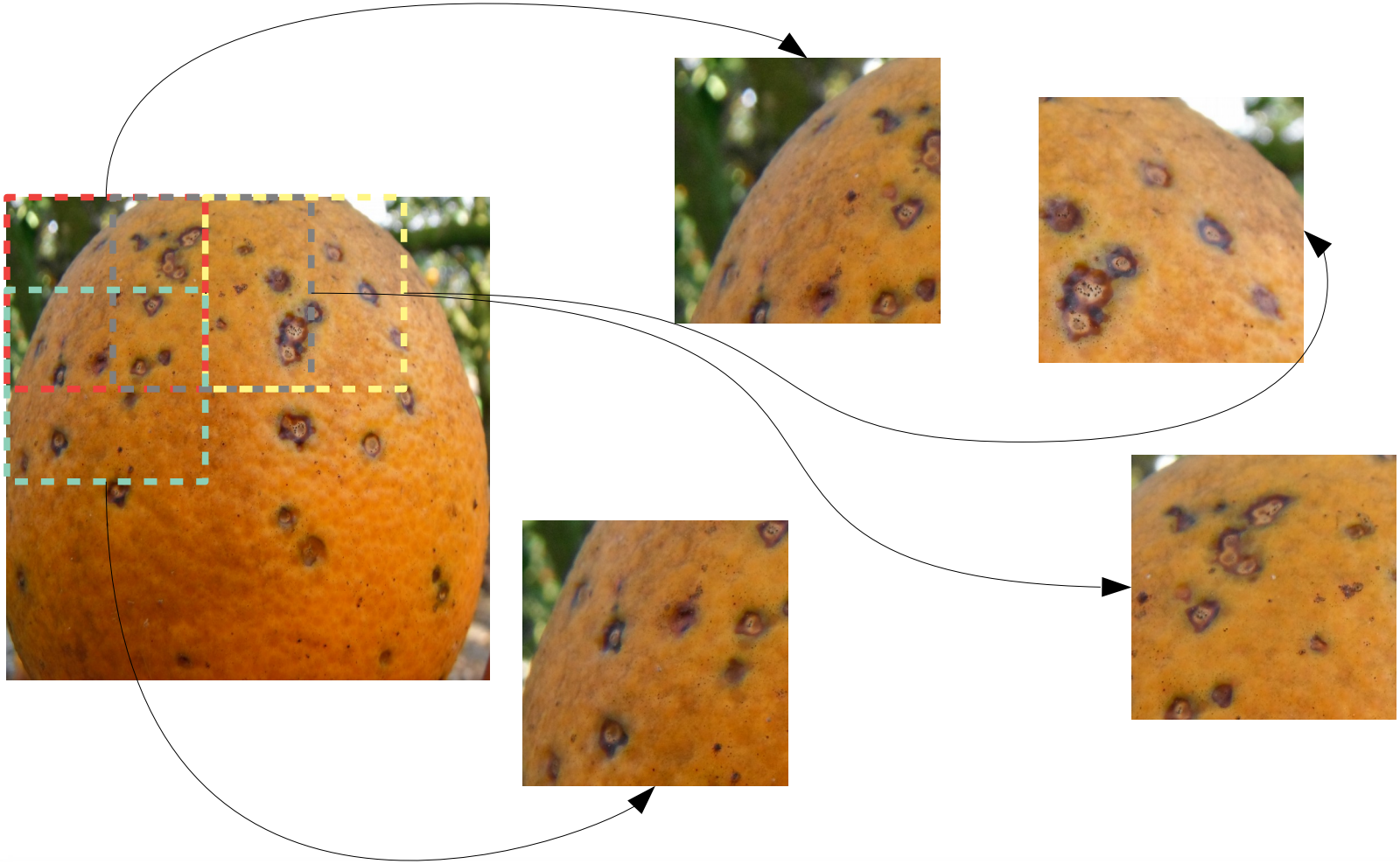
- Fruita/planta a moltíssimes malalties
- Mercat molt important
- Comerç entre tots els continents
- Riscos molt diferents a cada continent

Àfrica	↔	Europa (*Thaumatotibia*, *Pseudocercospora*, Tefrítidos, Moscas blancas, Thrips)
Amèrica	↔	Europa (Greenig, Tefrítidos, Ácaros)
Àsia	↔	Oceania
Àfrica	↔	Amèrica (Greening, Tefrítidos)
Europa	→	Àsia (Tefrítidos, Barreneta)

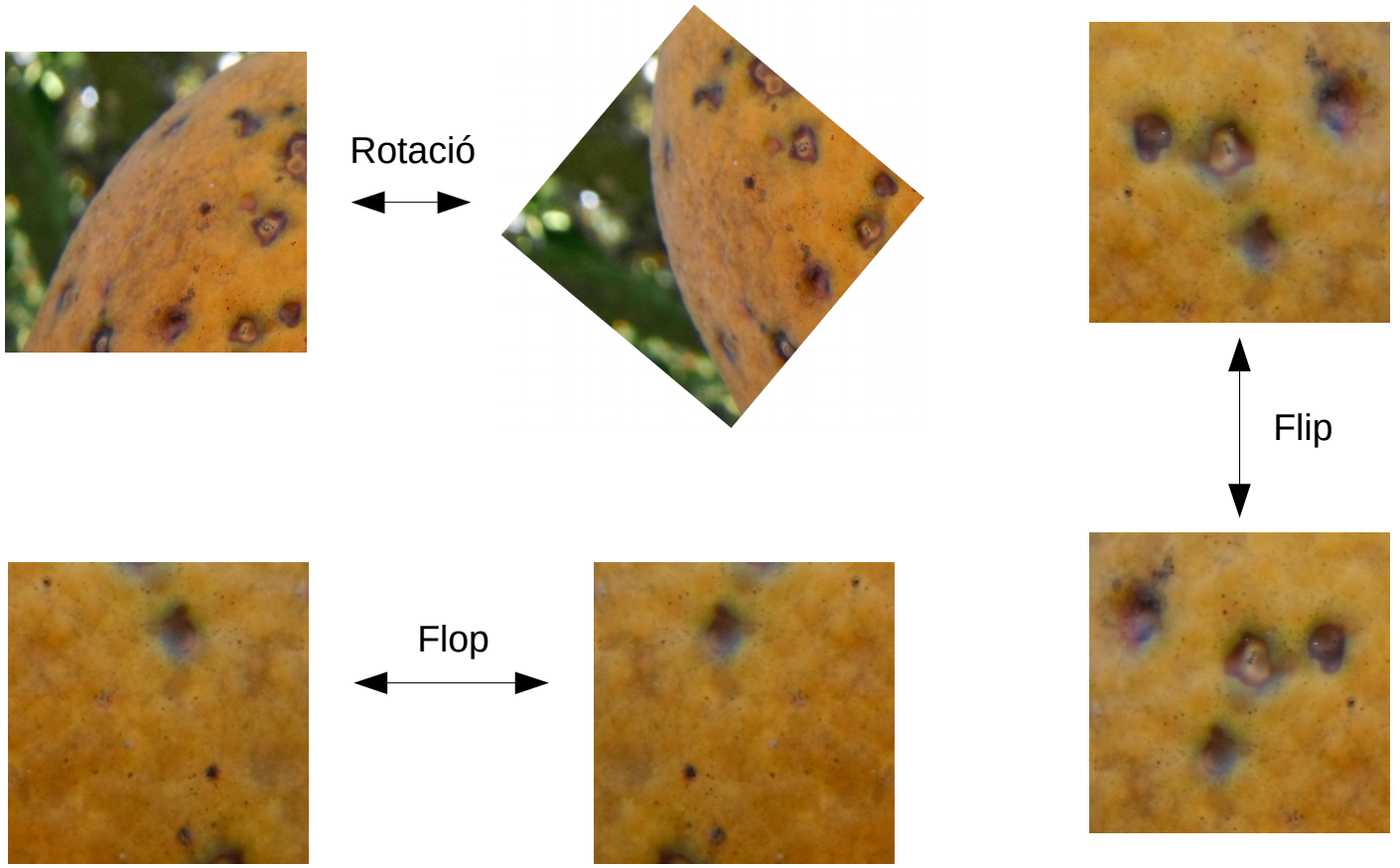
Malalties que s'han tingut en compte a l'estudi



Multi-split a les imatges originals

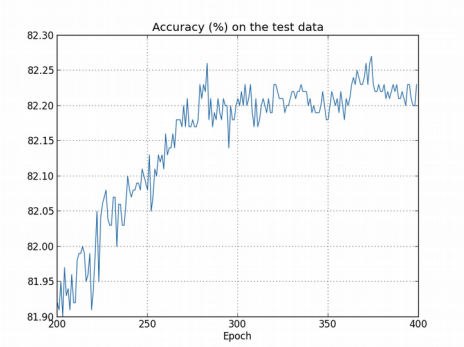


Transformacions



Problemes

- «DataSet» petit
- «Overfitting» → «Fine-Tunning»



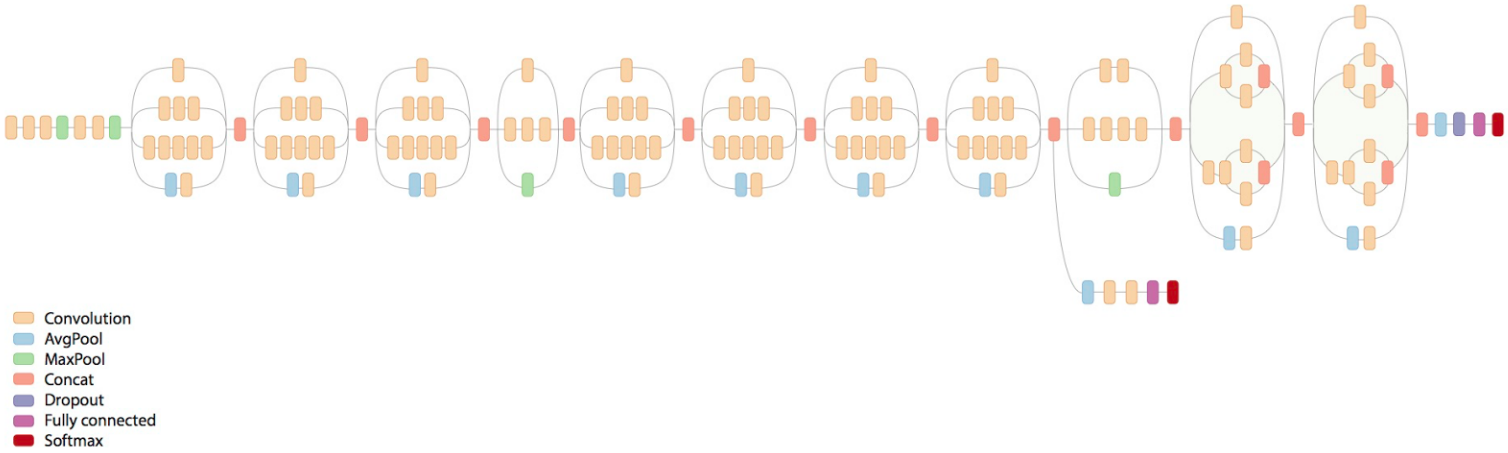
- Hiperparàmetre → mida del subsplit
  - 300 x 300 massa petit, moltes errades
  - 1000 x 1000 molt bona precisió, «dataset» encara més petit
  - Mida final → 500 x 500
- Altres transformacions:
  - «shift»
  - canvi de: il·luminació, color, brillo
  - ocultació de part de les imatges

Alternatives

- Models més clàssics: segmentació, extracció de característiques i classificació
- Molts estudis previs, cadascun ha d'ajustar les característiques del model al problema:
  - nombre de clústers
  - validació segons diversos algorismes de visió per computador
  - diversos mètodes de classificació amb els seus propis hiperparàmetres
- Xarxes neuronals profundes
  - mateix model per a diversos problemes
  - sempre el mateix «pipeline»
    - entrenament → [«cross-validation»] → test

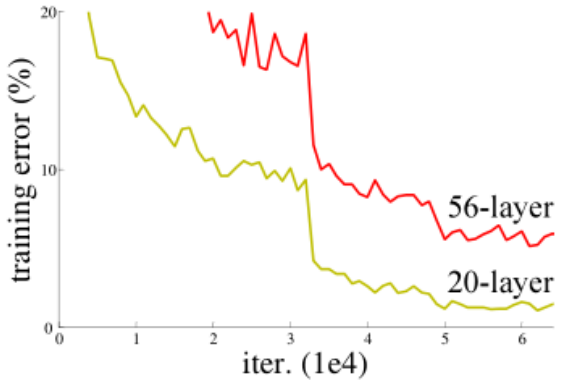
Xarxa Neuronal Profunda (CNN)

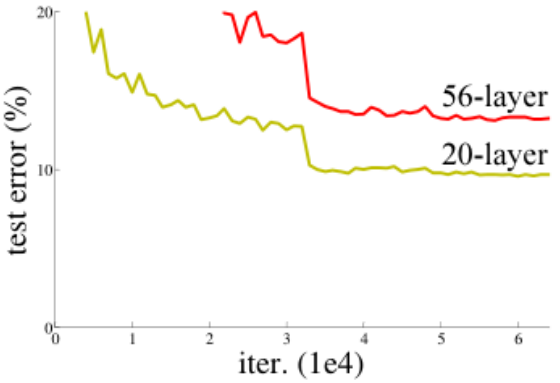
Article Stanford

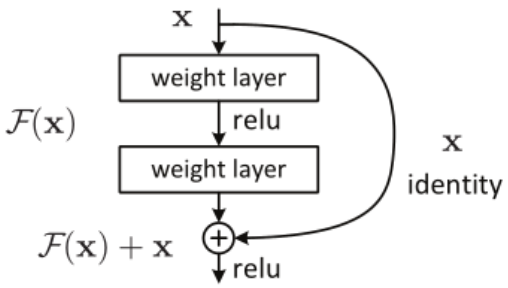


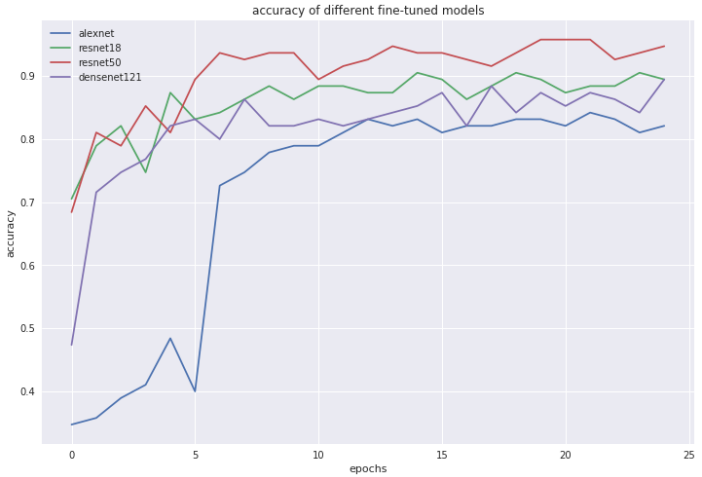
Dermatologist-level classification of skin cancer

RESNET



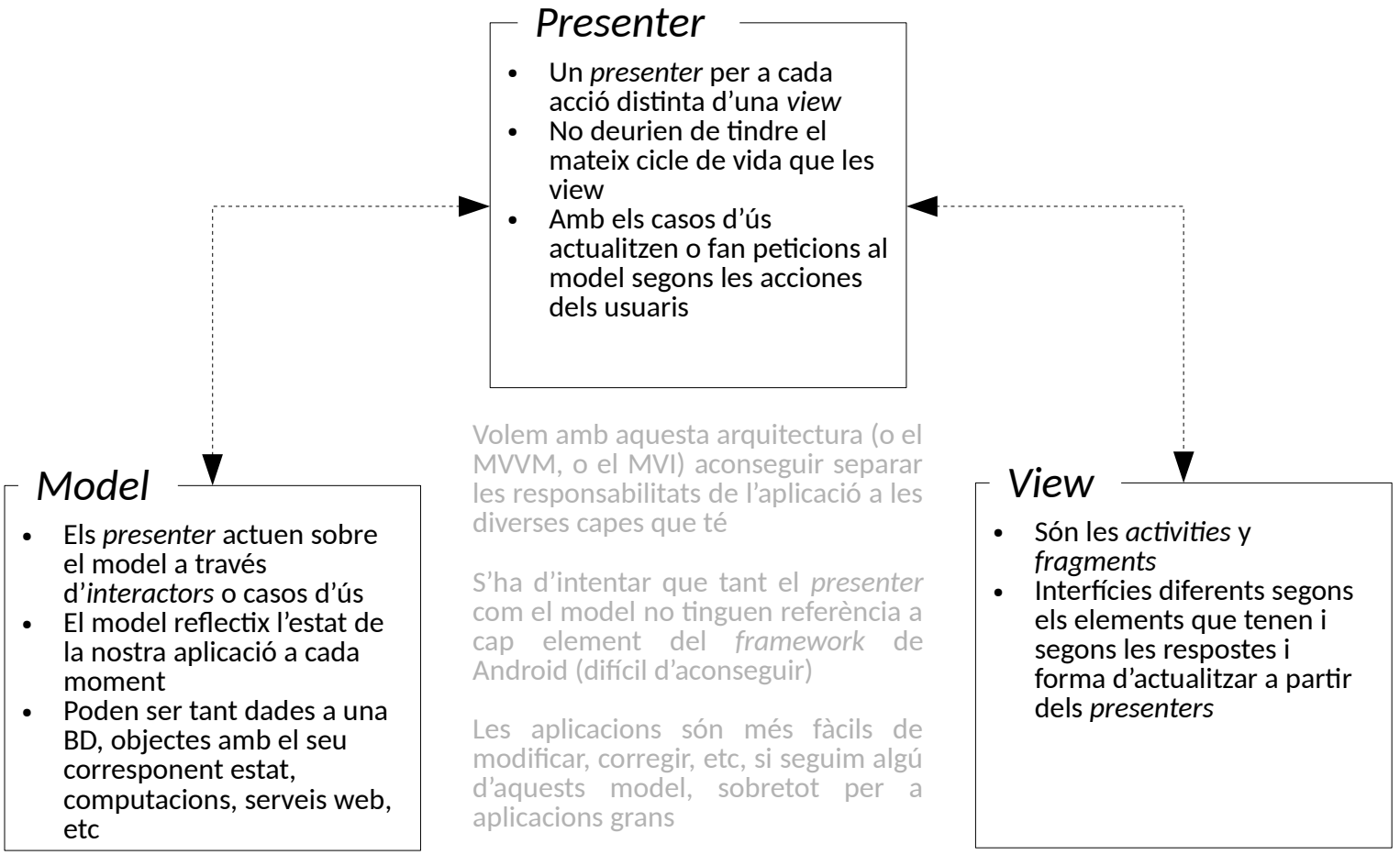






Article RESNET

Arquitectura



Volem amb aquesta arquitectura (o el MVVM, o el MVI) aconseguir separar les responsabilitats de l'aplicació a les diverses capes que té

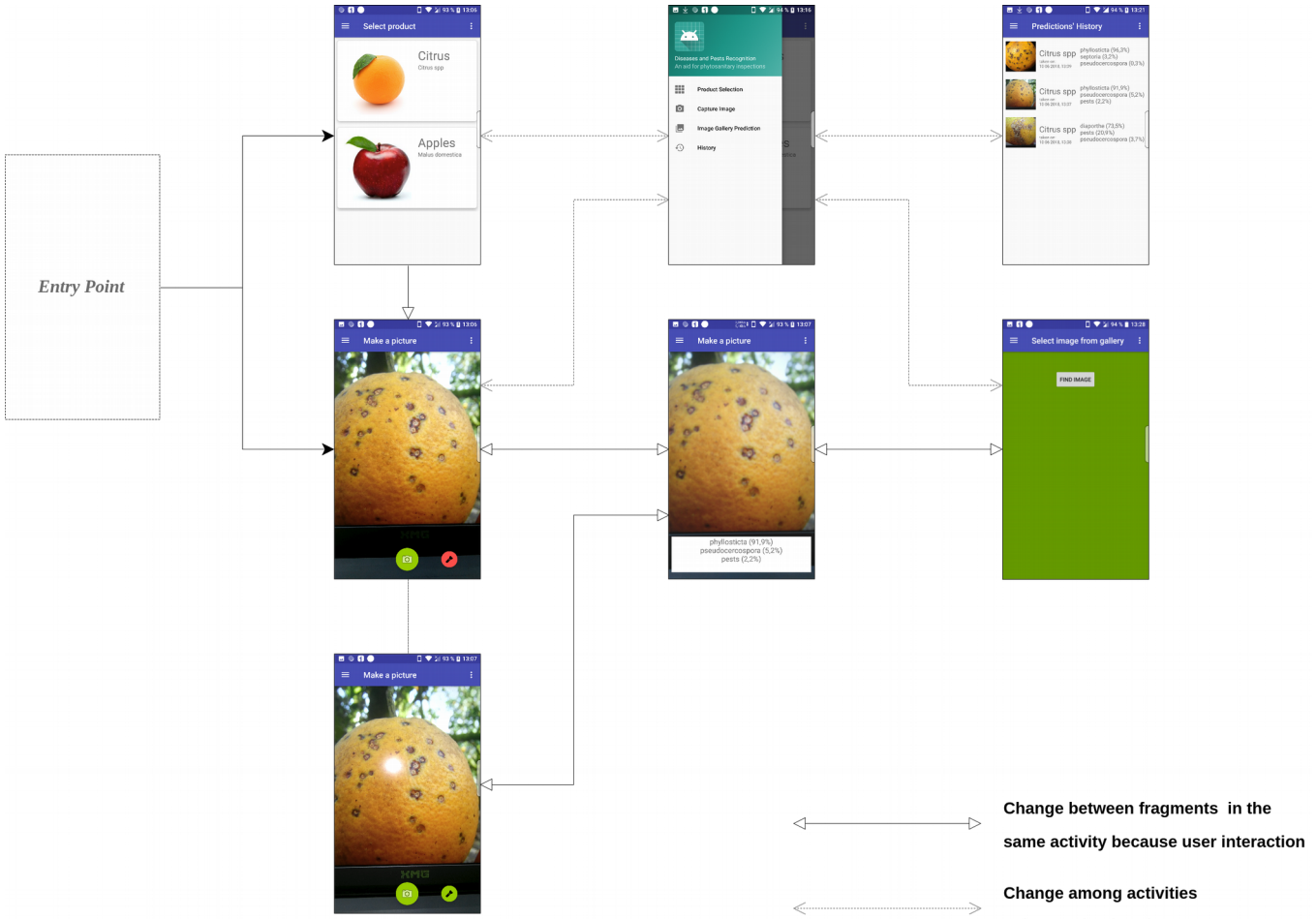
S'ha d'intentar que tant el *presenter* com el model no tinguen referència a cap element del *framework* de Android (difícil d'aconseguir)

Les aplicacions són més fàcils de modificar, corregir, etc, si seguim algú d'aquests model, sobretot per a aplicacions grans

Molt important, no tindre classes amb centenars de línies de codi (problema típic de les *activities*)

Aplicació mòbil (Android)

UI



Predictor → *TensorFlow Lite*

Una única línia a l'*script* de *gradle*

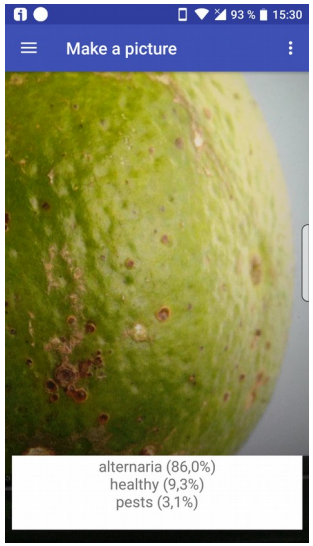
```
implementation "org.tensorflow:tensorflow-lite:+"
```

Instanciem el *Interpreter*, i només cal cridar al seu mètode *run*

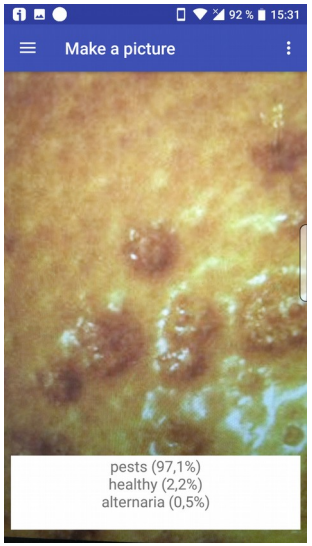
```
val interpreter = Interpreter(model: MappedByteBuffer)

Interpreter.run(img: ByteBuffer, probabilitat: Array<FloatArray>)
```

Extremedament més senzill que emprar *Caffe2* o *DL4J*

Resultats


Make a picture
alternaria (86,0%)
healthy (9,3%)
pests (3,1%)


Make a picture
pests (80,4%)
healthy (19,0%)
pseudocercospora (0,5%)


Make a picture
pests (97,1%)
healthy (2,2%)
alternaria (0,5%)


Select image from gallery
phyllosticta (96,3%)
septoria (3,2%)
pseudocercospora (0,3%)

Conclusions

Més importants

- Control d'errors a *Realm*
- Migrar de *PyTorch* a *TensorFlow* → molts problemes a la migració
- Provar models propis
- Entrenar models per a fruites
- Crear tests

Treballs futurs

- Entrenar model de cítrics amb més imatges
- Fer implementaió a iOS (swift/flutter/react)

Diferències primera/segona parta

- Més estudi
- Més implementació

Fi

Gràcies