XNAP-Segons la IA, quina edat aparentes?

En aquest projecte s'implementa varios models amb diferents datasets amb l'objectiu de predir l'edat d'una persona a partir d'una imatge. En aquest projecte s'ha treballat amb 3 datasets diferents:

- Appa_real: 7.5k imatges
- CACD (Cross Age Celebrity Dataset): 160k imatges de 2k celebrities diferents
- AFAD (Asiatic Face Age Dataset): 160k imatges de persones asiàtiques

Per cada dataset hi ha el seu folder personalitzat. Dins de cada folder hi trobem l'arxiu 'lectura_' + nom del dataset + '.py', 'model_' + nom del dataset + '.py', 'train_' + nom del dataset + '.py', 'main_' + nom del dataset + '.py', el requirements.txt i els arxius '.csv' que s'utilitzen per llegir les dades. En el cas del Appa_real, els arxius '.csv' no esta dins d'aquest repositori. Més endavant, s'explica com descarregar-se aquests datasets.
En el cas del dataset CACD també hi trobem el 'preprocessing_cacd'.

En aquest treball s'han fet diferents entrenaments i s'han dut a terme diferents proves ajustant constantment els hiperparàmetres, les arquitecures dels models i els datasets per tal d'intentar obtenir un rendiment óptim. Durant tot el procés, s'ha fet ús del Weight & Biases per tal de fer un seguiment de l'aprenentatge dels nostres models i visualitzar els resultats.

Estructura del codi

En cada folder hi trobem l'arxiu **'lectura_'+ nom del dataset + '.py'**. En aquest arxiu hi trobem una classe personalitzada per cada dataset on llegeix les imatges i les edats. En aquesta classe, se li pasa com a paràmetre el path del directori on es troben les imatges, el path d'on esta el csv amb el nom de l'arxiu de cada imatge i la edat corresponent i les transformacions en el cas que es vulguin aplicar, sinó no cal pasar-ho com a paràmetre.

També hi trobem el **'model_' + nom del dataset + '.py'**. En aquest arxiu python hi trobem la funció que crea el model. En cada folder hi trobem una configuració diferent ja que està personalitzat per cada dataset. En tots tres a la funció se li pasa el tipus de transfer learning (finetunning o feature extraction). S'utilitza transfer learning en cada un amb l'arquitectura resnet34 pre-entrenada.

En cada folder veiem un **'train_' + nom del dataset + '.py'** En aquest arxiu està la funcio train. Se li pasa com a paràmetres el model creat, els dataloaders, la loss function, l'optimizer, el número de épòques que es vol entrenar el model, el nom del projecte wandb i el nom de l'execució i per últim el device. Aquesta funció entrena el model amb el dataloader del train, l'evalua amb el dataloader del validation i retorna el state del millor model trobat i les losses.

Hi ha un arxiu **'main_' + nom del dataset + '.py'** en cada folder. En aquest arxiu, s'inicia sessió al wandb, es crea un projecte si es vol crear, es crida a la lectura del dataset ('lectura_' + nom del dataset + 'py.') i es creen els dataloaders, es crea el model i es defineix la loss function i l'optimizer. Un cop tot definit i creat, es crida al train perque el model s'entreni i es guarda el state del millor model trobat, retornat per la funció train, a un arxiu '.pth'.

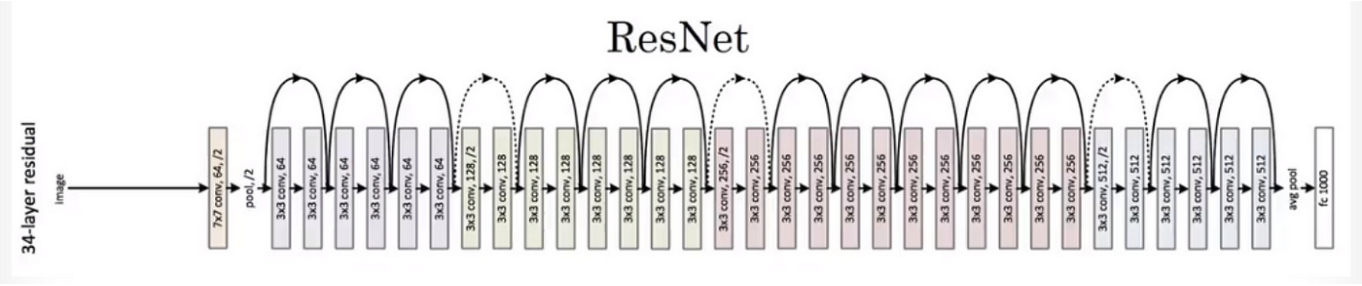
Per últim, hi ha l'arxiu **'test_' + nom del dataset + '.py'** on creem el model cridant a la funció de l'arxiu 'model_' + nom del dataset + '.py' i al model li fem un load de l'arxiu creat '.pth' on estan els paràmetres i els seus valors del millor model guardat. Després es carga la imatge que es vol testejar, la transformem en tensor i en treiem el seu output.

Pel que fa al **preprocessing_cacd.py**, llegim cada imatge del dataset i el passem per un detector de cares. Si aquest detector de cares detecta 1 cara, aleshores aquesta es retalla i s'afegeix com a imatge a un nou directori que es crea. Si no en detecta cap cara, més d'una o hi ha un error, aquesta imatge no s'afegeix. Per tant, després de fer això, es creen 3 datasets, un per train, un per valid i un per test, amb només el nom del fitxer imatge i l'edat de les imatges que s'han guardat al nou directori.

Execucions

A continuació s'explicaran les diferents execucions i ajusts que s'han fet per arribar a un model final.

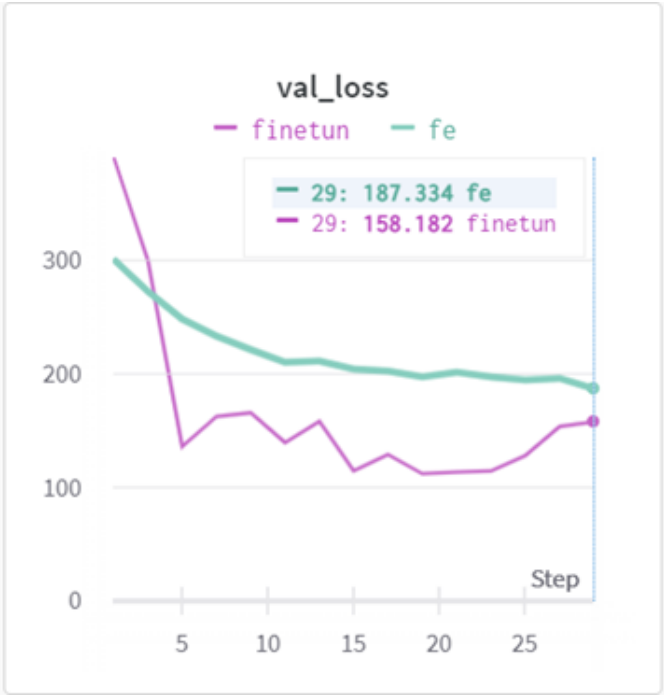
Primer de tot, aquesta és l'arquitectura de Resnet34 que utilitzem en pràcticament totes les execucions i és l'arquitectura dels nostres models finals:

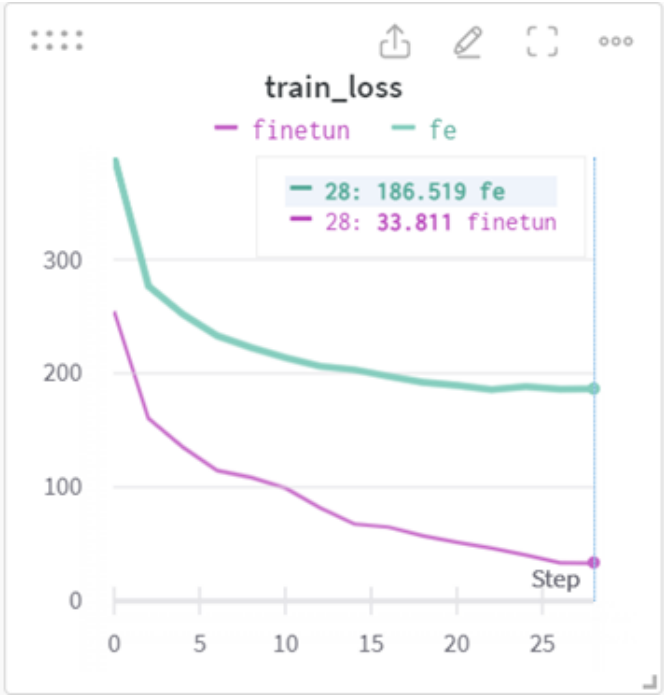


El model consta de 34 cappes residuals, on es divideixen en diferents BasickBlocks (de diferents color) i dins de cada bloc, apart de les convolucions, també hi trobem capes de batch normalization i funcions d'activació Relu. L'arquitectura acaba amb un AdaptiveAveragePool per reduir el tamany espacial dels feature maps, i per últim una capa FullyConnected, on, en el nostre cas, li demanarem un output de 1 ja que estem fent regressió.

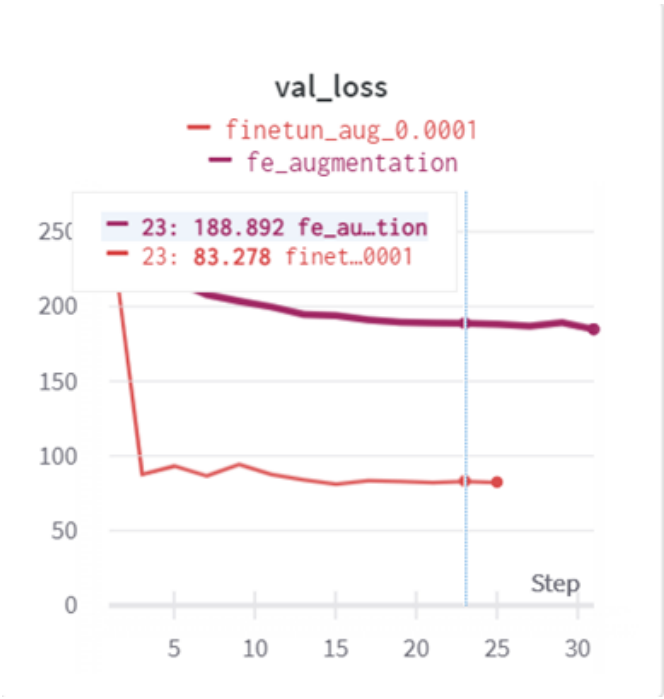
Appa Real

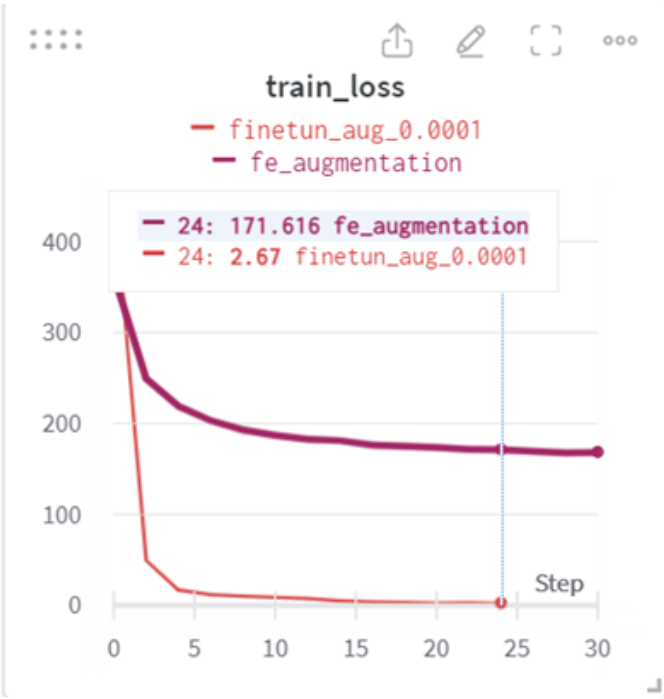
Aquest dataset tenia 4k d'imatges en el train i 1.5k en el validation. En la lectura de les dades es van aplicar unes transformacions. Primer un resize i un centercrop perque totes les imatges tinguessin el mateix tamany i després de crear el tensor de la imatge es normalitzava. Es van dur a terme dos execucions fent ús de l'arquitectura resnet34 i la loss MSE. D'una banda es va fer feature extraction, on congelem totes les capes menys la última, i d'altra banda finetunning on cap capa esta congelada. Els resultats van ser els següents:



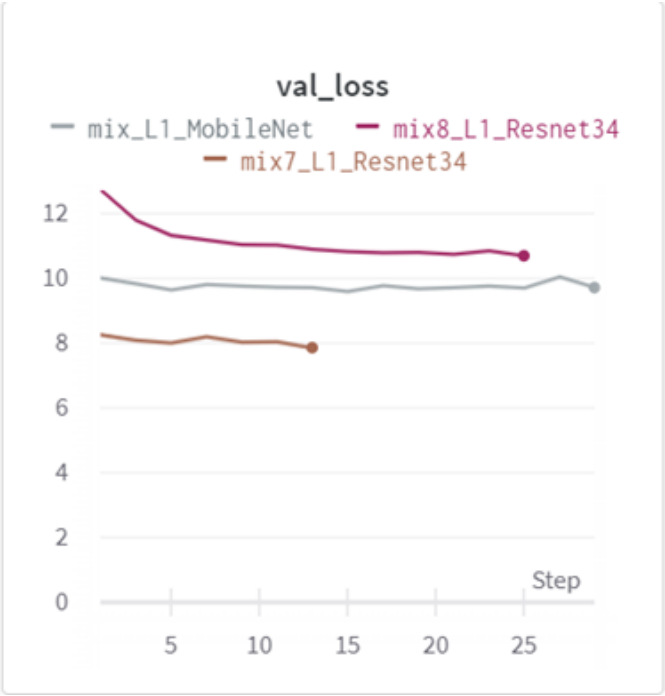


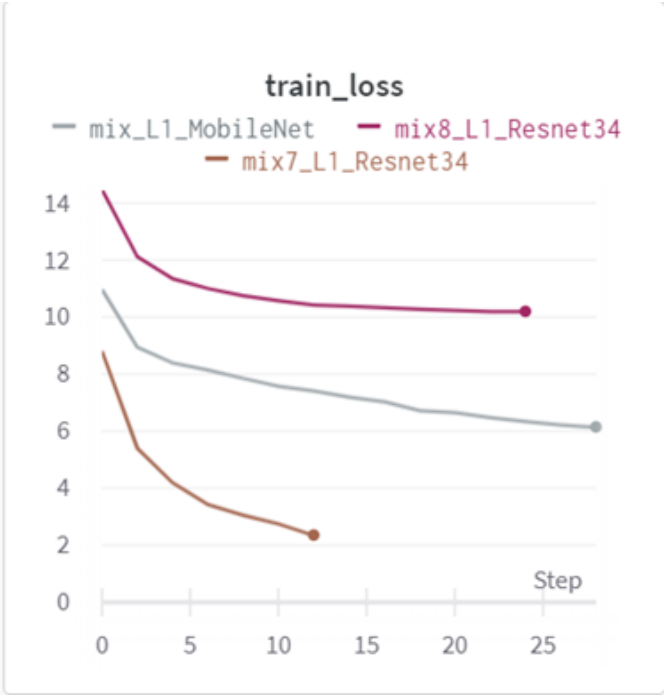
En la imatge veiem la loss del train i del validation en les diferents execucions. L'eix de les x, son les èpoques x2, és a dir, en aquest cas s'han fet 15 èpoques. Com es veu a la imatge, obtenim un model amb overfitting clar pel que fa al finetunning i un rendiment poc óptim amb el feature extraction. Aquest overfitting hem cregut que pot ser donat a la poca quantitat de dades. És per això que fem un data augmentation i passem de 4k a 12k d'imatges en el train. Per fer el data augmentation es fa per cada imatge un random rotation i una transformació de color on apliquem diferents valors de brillantor, contrats, saturació,.. Executem igual que abans ajustant ara si el learning rate en el finetunning a 0.0001.



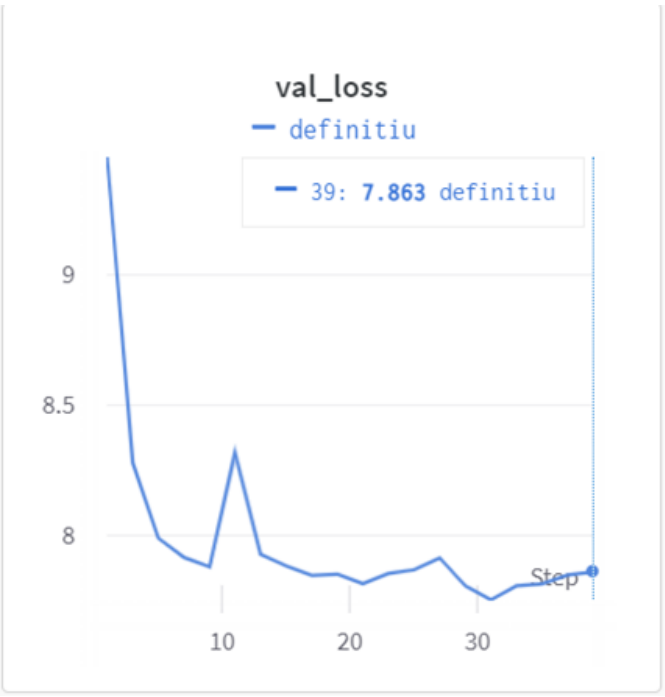


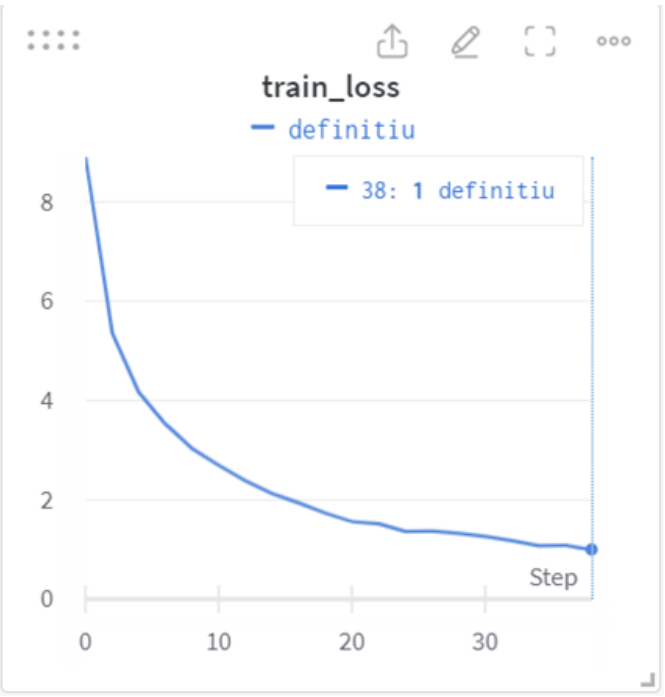
Veiem que el rendiment del finetunning en el train és molt bo pero hi ha un overfitting clar. Aleshores el que vam pensar va ser fer feature extraction pero descongelant alguna capa més buscant un terme mig entre finetunning i feature extraction. També vam canviar la loss a L1 i vam provar una arquitectura mobilenetV2, una menys complexa, per veure com rendia el model.





En aquest gràfic veiem com descongelant també l'últim BasicBlock de l'arquitectura resnet34 (mix7_L1_Resnet34) obtenim un rendiment millor que només descongelant el average pooling i la última capa lineal o utilitzant una altra arquitectura. Un cop veiem el rendiment de cada model, concluim que el millor model obtingut és el model resnet34 descongelant l''últim BasicBlock. Executant-lo amb 20 èpoques obtenim un rendiment óptim en el train però un overfitting clar en el validation que com a millora s'hauria d'intentar reduir.





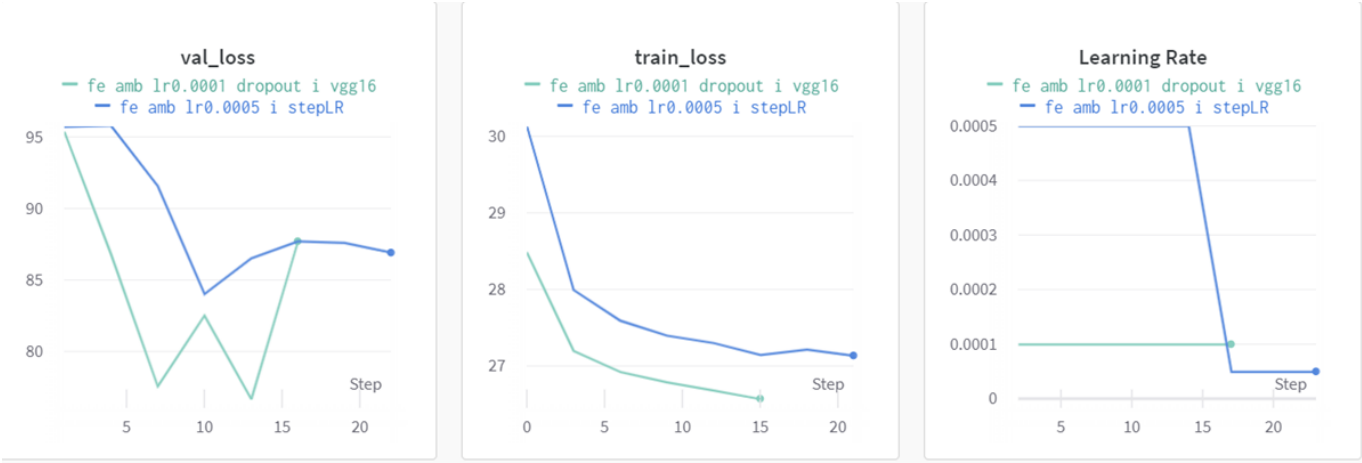
AFAD

Amb el dataset dels asiàtics, llegim les dades i apliquem transformacions semblants a les esmentades anteriorment. Fem el data loader amb un batch size de 128. Les dades que tenim son 119k en el train i 13k en el validation. La primera execució que vam fer va ser un feature extraction i vam veure que la los del train s'estancava i a mesura que pasaven les épocques no millorava. Una solució que vam pensar va ser aplicar una

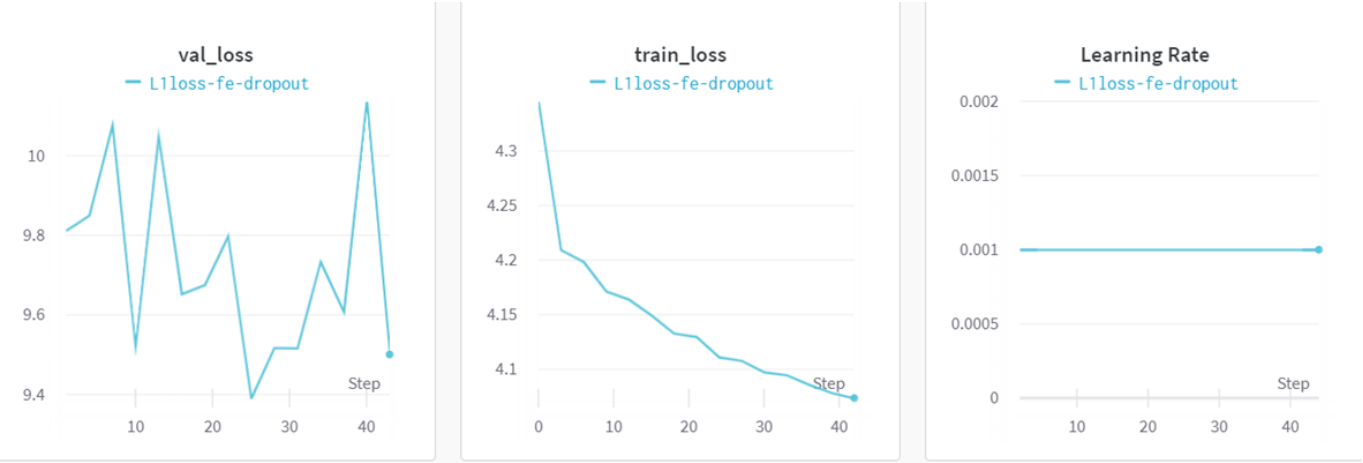
degradació al learning rate de tal manera que potser pogués sortir d'aquell mínim que creiem que s'havia quedat i trobar un óptim.

Veiem que desrpés de 8 époques aprox degradem el learning rate i la loss del validation que havia baixat i pujat en picat torna a descendir. Tot i així, veiem un overfitting clar comparant una losa de 27 del trait amb 86 del valid.

Per això, vam provar de canviar el model al vgg16 i provar de fer ús de la eina drop-out, on desactivem aleatoriament algunes neurones de la última capa. Amb això intentem reduir el overfitting però no aconseguim aquest objectiu.



Per últim, canviem la loss a L1 com hem fet anteriorment amb el dataset Appa Real. Aquest és el nostre model final per aquest dataset on fem un feature extraction amb la Resnet34 fent un drop-out en la capa fully Connected final. Concluim que no es un rendiment ideal ja que ens trobem overfitting i l'error es significant.

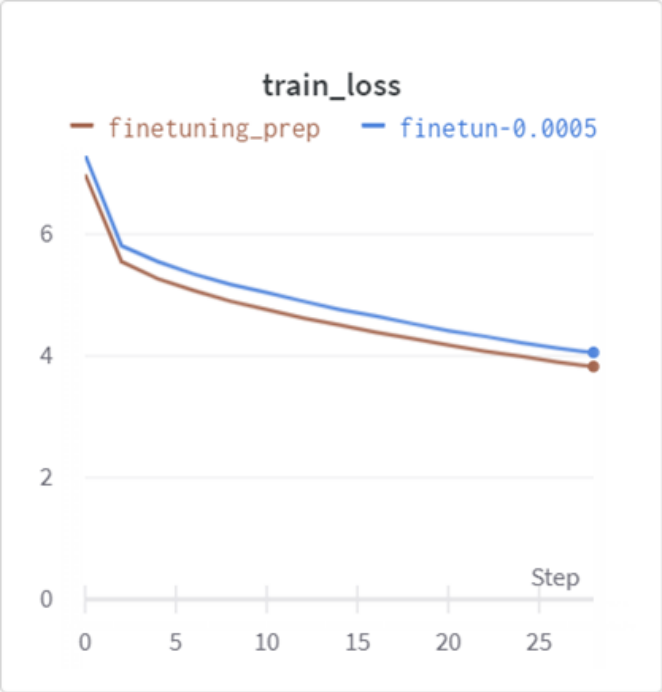


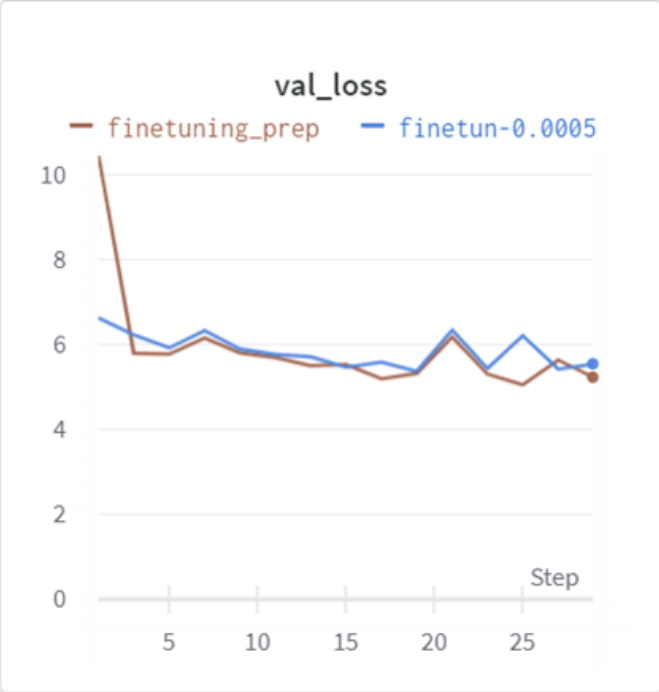
CACD

Per últim, l'últim dataset que hem tractat ha sigut el dels famosos on contem 115k per el train i 12.7k pel validation. En aquest dataset també hem aplicat transformacions, hem utilitzat el model resnet34 i hem partit ja inicialment amb la L1 com a loss.

En la primera gràfica es veu el rendiment del model entrenat amb finetunning i un valor de learning rate 0.0005. Aqui cal destacar que no en trobem tant overfitting com hem trobat als altres models.

Per últim, hem provat un preprocessat d'aquestes dades per a veure si obteníem un rendiment encara millor. Hem fet ús de les llibreries opencv i dlib per a que de les imatges en detectes les cares de la foto i si en detectava, la retalles i crees una altra foto retallada amb només la cara, i aquesta seria la que pasariem al model. Veiem que aquest canvi no ha sigut molt significatiu però ens ha servit per idear alguna proposta de millora pel futur.





Resultats finals

Aquests son els resultats finals dels millors models obtinguts amb els diferents datasets. En tots els casos sempre hi ha un marge de millora.

	Appa-Real	AFAD	CACD
Train_loss	1	4.07	4.05
Valid_loss	7.86	9.5	5.54

Tests

Després d'extreure els models ideals, provem a fer algun tests amb els dos millors models d'una foto d'un dels contribuidors d'aquest treball:

D'aquesta foto, el model Appa-Real predeix que en té 14 anys. Mentre que el CACD diu que en té 33 anys. Aquesta diferència d'edat predita creiem que pot ser deguda a que en dataset dels famosos conté imatges de famosos on ells estan maquillats i ben pulits i aparenten menys edat de la que tenen, per això, una persona normal el model pot creure que té més anys dels que té.

També hem volgut fer una prova amb una noia per veure quin seria el resultat:

El model Appa-Real predeix que en té 15 anys mentre que el dels famosos diu que en té 32. En aquest cas, es repeteix una tendència que també es pot observar i que ja hem comentat en el test anterior.

Example Code

En aquest projecte es treballa amb 3 datasets. Per descarregar-se cada un d'ells, ho expliquem a continuació:

- Appa-Real: *url= 'http://158.109.8.102/AppaRealAge/appa-real-release.zip' datasets.utils.download_and_extract_archive(url, '../AppaRealAge')*
- CACD: https://bcsiriuschen.github.io/CARC/ en aquest enllaç hi ha l'opció de descarregar-se un .tar.gz amb totes les imatges
- AFAD: https://github.com/John-niu-07/tarball en aquest github esta el dataset separat per peces i utilitzant shell script amb l'arxiu restore.sh s'ajunten totes les peces per crear el dataset original.

Tots tres per pasar-los a la màquina virtual i fer-los servir en Azure, es va fer servir el programa FileZilla, que ens permetia conectar-nos a la màquina azure i pasar arxius del local a la màquina.

Per executar els arxius de cada folder (dataset), primer cal instal·lar-se les llibreries corresponents amb les seves versions.

```
pip install -r requirements.txt
```

Un cop instalades, activem conda.

```
conda activate base
```

I després ja es poden executar els arxius corresponents.

```
python nom_arxiu.py
```

Alguns arxius python estan configurats amb una ruta als datasets específica. Aquestes variables amb rutes personalitzades estan marcades amb un comentari '#CHANGE FOR YOUR CASE'. Un cop descarregats els datasets, canvieu les rutes per el vostre cas.

Contributors

Gabriel Gausachs Fernández de los Ronderos 1604373@uab.cat

Arnau Gómez 1601488@uab.cat

Xarxes Neuronals i Aprenentatge Profund Grau de Data Engineering, UAB, 2023