UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria electrònica industrial i automàtica

MACHINE LEARNING AMB DADES D'OCELLS

Memòria i Annexos

Autor: Joan Hidalgo López
Director: Gerard Escudero Bakx

Convocatòria: Juny 2022

Resum

Aquest projecte tracta de aplicar un algoritme de Machine Learning el qual utilitzant eines de processament de imatges detecti i classifiqui àudios de ocells. Assolit aquest objectiu es treballa en fer un estudi sobre la creació d'una base de dades òptima per la implementació del algoritme creat depenent el número de especies a classificar.

Aquest treball inclou diferents experiment, el primer es tracta de la creació d'un model funcional per a la classificació de 4 especies i el segon del estudi de la base de dades, utilitzant el model funcional adquirit en el experiment anterior. L'algoritme del model creat es tracta de EfficientNet el qual es de menor dimensió i més ràpid que les Convolutional neural networks(CNNs) convencionals.

El model predictiu en les 4 especies amb un epoch de 30 ha generat una precisió del 98%, classificant la totalitat del conjunt de validació.

Índex

1. Introducció

1.1. Objectius

Aquest treball consta de 2 objectius principals, el primer de la implementació d'un model que permeti la classificació de dades de ocells y el segon del estudi de la base de dades per aconseguir un bon model. Aquesta feina consisteix en descobrir el format que han de tenir les dades i la mida del conjunt depenent de la quantitat de classes, on s'ha utilitzar les corbes d'aprenentatge per determinar-ho.

1.2. Abast del treball

Les tasques requerides per la realització dels diferents experiments són les següents:

- **Obtenció de les dades**

L'obtenció de les dades que el algoritme ha d'utilitzar, en específic es tracta de arxius de àudios.

- **Preparació del conjunt de dades**

Posterior a l'obtenció del conjunt dades s'inicia el procés de preparació d'aquets.

Un primer pas es l'etiquetatge d'aquest fitxers d'àudios descarregats, ja que l'aprenentatge supervisat requereix que el model pugui detectar les diferents classes compreses en el conjunt de dades. En aquest cas s'ha fet amb la creació de carpetes i subcarpetes on els àudios anaven dintre de la carpeta que corresponia a la seva espècie.

Després aquests àudios s'havien de transformar en imatges ja que els mètodes de Machine Learning no treballen directament amb arxius d'àudios. Concretament s'han obtingut els gràfics en forma de espectrogrames de les mostres mitjançant programació en Phyton i es desaven en les carpetes corresponents per la seva posterior utilització. On a més es classifiquen en conjunt de entrenament de test i de validació. El primer conjunt es el que s'utilitzarà per configurar el model de classificació, mentre que el test i la validació s'utilitzen per determinar si el model predictiu creat classifica correctament.

Un aspecte a tenir en compte es que les dades de validació i de test no han de formar part de les dades utilitzades al entrenament per fer una correcta avaluació del model de classificació.

Dintre d'aquest procés de la creació d'espectrogrames hi existeix un treball de filtratge ja que es realitza un filtratge digital de freqüències per obtenir una forma d'ona dels ocells més neta.

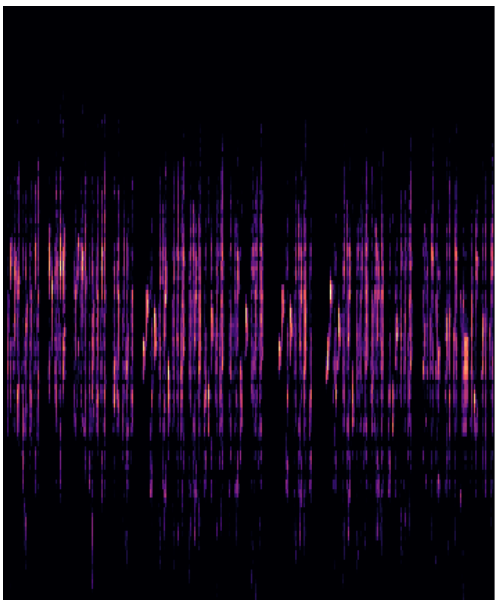


Figura 1.1. Exemple d'espectrograma creat corresponent a la espècie d'ocell Barn Swallow.

- **Creació del model**

  Amb el pas de la preparació de dades complert el següent pas es escollir una algoritme que s'adapti al nostre objectiu. Hi existeixen diferents tècniques de classificació com K-Nearest Neighbors, Support Vector Machines (SVM) o Neural Networks(NN) entre d'altres.

  En aquest cas d'estudi s'ha optat per la utilització de Convolutional Neural Networks(CNNs) en específic per la variació de EfficientNet. Aquesta idea ha sigut extreta dels guanyadors de diferents concursos on l'objectiu era classificar àudios d'ocells que utilitzaven CNNs o RNN i he escollit el model de CNN per la seva facilitat de entrenament respecte els models de RNN (1)(2).

- **Avaluació i millora del model**

  Si el comportament no dona el resultat desitjat s'ha de revisar els punts anteriors, com el augment de la base de dades. A més de modificar inputs de la funció del model.

El treball consisteix en trobar una precisió acceptable per obtenir un model funcional amb el que treballar per fer l'estudi de la dimensió de la base de dades.

- **Creació de corbes de aprenentatge**

Amb el model creat es comença a fer execucions de programes amb diferents numero de classes i número de dades, per agafar punts d'interès.

1.3. Dataset

Les dades s'han obtingut a través de la pàgina web de Xeno Canto, on hi trobem un gran repertori d'àudios d'ocells de diferents especies en format mp3. En la pàgina hi trobem el nom de l'espècie, la localització de la gravació, llicencies aplicables i l'autor del arxiu entre altres dades.

El copyright de cada fitxer d'àudio pertany al usuari que ha donat el seu fitxer a Xeno Canto. Reconeixements a xeno-canto(3),per la base de dades.

1.4. Model

EfficientNet és un tipus de Convolutinal Neural Network, una xarxa neuronal d'aprenentatge profund que és artificial. S'utilitza en visió per ordinador i reconeixement d'imatges (4).

Aquest ha sigut el model utilitzat per fer la classificació de els àudios dels ocells.

El model ha sigut desenvolupat en llenguatge Phyton sobre la plataforma de GoogleColaboratory (5).

1.5. Motivació

El desenvolupament del projecte es duu a terme per conèixer les tècniques i l'us del Deep Learning creant d'aquesta manera un algoritme que permeti ser d'ajut en l'estudi de diferents especies.

2. Estat de l'art

2.1. Estat global

Ens trobem en el segle 21, moment en el qual utilitzem assistents per veu com Alexa, on empreses com Tesla desenvolupen noves tecnologies com cotxes autònoms. Vivim en un món envoltat de Deep Learning. Des de les recomanacions de Spotify al com el reconeixement d'imatges.

Els avenços vénen acompanyats amb alguns temors, com l'amenaça de la seguretat laboral. Un estudi de la BBC prediu una automatització de molts llocs de treball en un futur (6).

Els científics i els enginyers són els encarregats d'administrar i supervisar aquestes dades. El Deep Learning no tracta només de programar, l'estadística, la probabilitat, el càlcul i l'àlgebra son coneixements fonamentals per treballar en aquest àmbit.

2.2. Que es el Deep Learning?

Per explicar que es el Deep Learning partim primerament del concepte de intel·ligència artificial, que es consideraria un programa capaç de sentir, raonar i adaptar-se. D'aquest conjunt en podem extreure un subconjunt anomenat Machine Learning, que serien els algoritmes específics que en milloren amb la rebuda de dades amb el pas del temps.

El Deep Learning es un subconjunt del Machine learning, que alhora es un subconjunt de la intel·ligència artificial. Que es essencialment una xarxa neuronal que intenta emular percepció visual humana, permetent-li aprendre de una gran base de dades (7).

El Deep learning elimina parts de el pre-processament de dades, permetent utilitzar dades com text i imatges ,i extreure característiques sense la dependència humana. Com exemple els algoritmes de Deep Learning poden determinar quin es característiques son més important alhora de fer una classificació mentre que en el Machine Learning això s'especifica manualment per un humà.

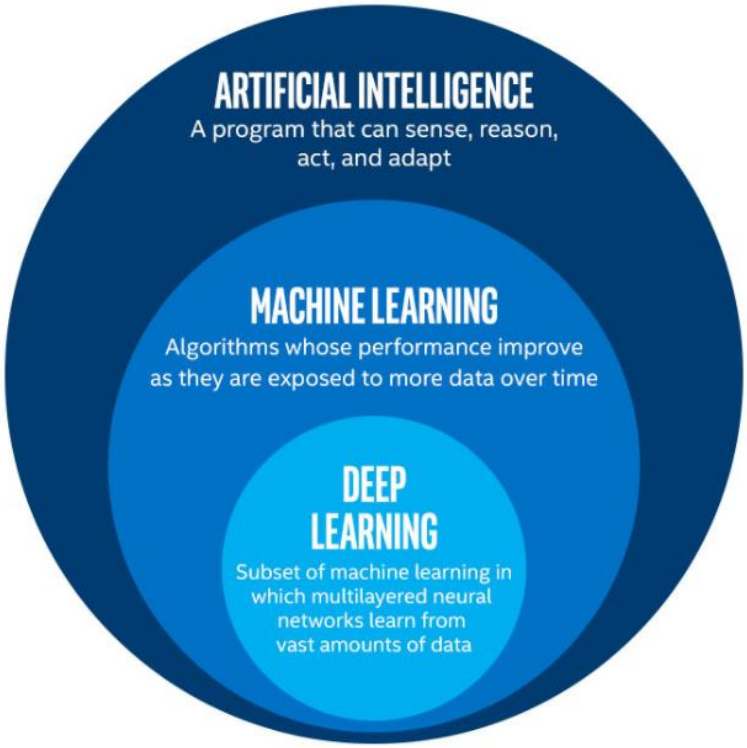


Figura 2.1. Esquema de la Inteligencia Artificial (8).

El Deep Learning descobreix una estructura intricada de gran conjunt de dades fent ús de l'algoritme de retropropagació en la capa anterior. Gràcies a aquestes xarxes profundes s'han produït avenços en el processament de imatges i audio.

La retropropagació és el mètode més habitual per el entrenament de les arquitectures tipus multicapa. Les arquitectures multicapa poden ser entrenades mitjançant descens de gradients estocàstic. El procediment de la retropropagació consisteix en calcular el gradient de la funció objectiu respecte els pesos de un grup de mòduls. La idea més importat es que el gradient es pot calcular treballant cap enrere des de el gradient a la sortida del mòdul. Per passar entre les capes, es calcula la suma ponderada de les seves entrades de la capa anterior i passa el resultat a través d'una funció no lineal (9).

2.3. Aprenentatge supervisat

https://spartanhack.com/machine-learning-y-deep-learning-todo-lo-que-necesitas-saber/

Es tracta d'un tipus d'aprenentatge , on els algoritmes utilitzen dades prèviament etiquetades, tenint l'objectiu que donant una entrada retorni una etiqueta de sortida. L'algoritme es capaç de aprendre de les decisions anteriors i així predir el valor de sortida. Aquests valors poden ser numèrics si es tracten de algoritmes de regressió o categòrics si son de classificació (10).

A continuació, es presenta un exemple sobre l'aprenentatge supervisat:

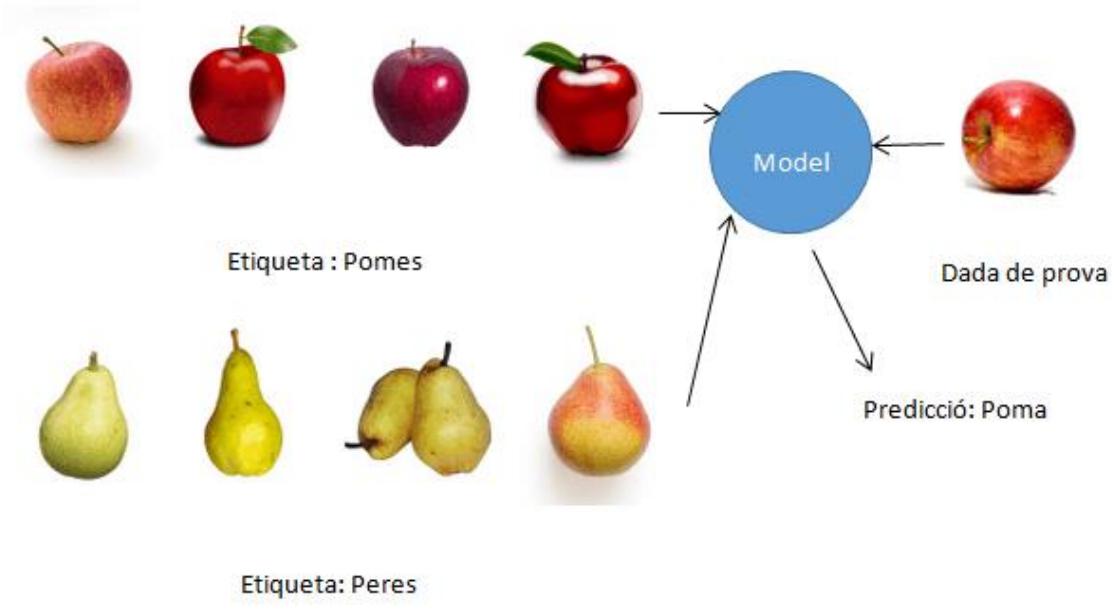


Figura 2.2. Esquema d'aprenentatge supervisat per la classificació de pomes i peres.

Primerament tenim les dades de entrada, les quals son imatges de pomes i peres que tenen la seva etiqueta corresponent.

Amb aquestes imatges el programa crea l'algoritme i va modificants els pesos per millorar la precisió i reduir l'error del mateix. Per el correcte càlcul dels pesos el algoritme calcula un gradient per descobrir quina quantitat variaria l'error per un increment lleuger del pes.

Al acabar l'entrenament s'analitza el rendiment amb un conjunt de prova. Per analitzar la capacitat de produir respostes coherents a imatges que no ha vist amb anterioritat.

2.4. Xarxes neuronals convolucionals

Les Xarxes neuronals convolucionals (CNNs) estan dissenyades per processar dades que es presenten en forma de múltiples matrius, per exemple una imatge en color composta per tres matrius 2D que contenen intensitats de píxels en els tres canals de color. Moltes modalitats de dades tenen forma de matrius múltiples: 1D per a senyals i seqüències, inclòs el llenguatge; 2D per a imatges o espectrogrames d'àudio; i 3D per a imatges de vídeo o volumètrics.

Una xarxa neuronal convolucional (CNN) és un algoritme d'aprenentatge profund que pot agafar una imatge d'entrada, assignar importància (pesos aprendre i biases) a diversos aspectes/objectes de la imatge i ser capaç de diferenciar-los els uns dels altres. El pre-processament necessari en una CNN és molt inferior en comparació amb altres algorismes de classificació. Mentre que en els mètodes primitius els filtres estan dissenyats a mà, amb prou formació, CNN tenen la capacitat d'aprendre aquests filtres/característiques. L'arquitectura d'una CNN és anàloga a la del patró de connectivitat de les neurones del cervell humà i es va inspirar en l'organització de l'escorça visual. Les neurones individuals responen als estímuls només en una regió restringida del camp visual coneguda com a camp receptiu. Una col·lecció d'aquests camps se superposen per cobrir tota l'àrea visual (11).

La arquitectura consta d'una part convolucional i una segona part que pot ser de regressió, classificació entre altres, aquesta segona capa l'usuari l'adequa depenent la feina que vulgui realitzar. L'objectiu de la part convolucional es extreure les característiques de cada imatge reduint la seva mida inicial mitjançant una sèrie de filtres, per finalitzar aquesta primera part els resultats obtinguts se concatenen en un vector de característiques anomenat codi CNN. En la segona part s'introdueix com entrada el codi obtingut a la primera part a un perceptró multicapa(MLP) (12)

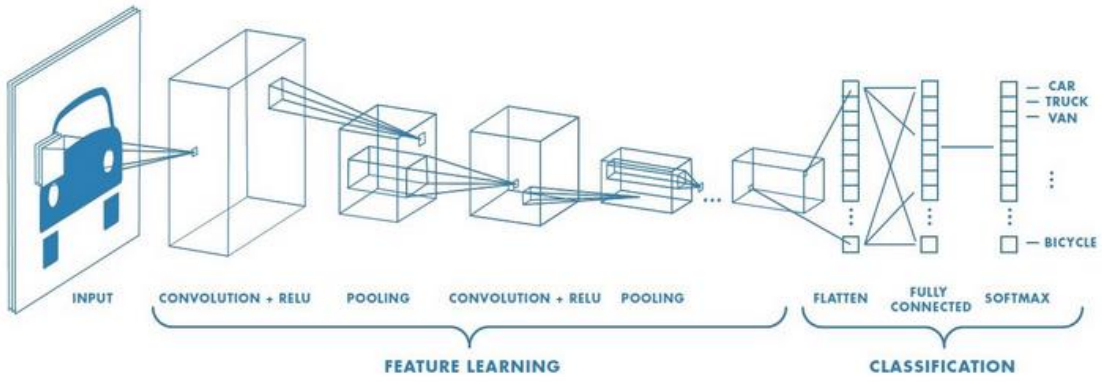


Figura 2.4. Esquema CNN (11).

2.5. Xarxes neuronals recurrents

Les Xarxes Neuronals Recurrents o en anglès Recurrent Neural Networks (conegudes amb l'acrònim RNN), són un tipus de xarxa que s'utilitzen per analitzar dades de sèries temporals permetent tractar la dimensió del temps. Quan es va introduir la retropropagació, l'ús més interessant va ser entrenar aquest tipus de xarxes neuronals. Les RNN són molt potents per tot allò que té a veure amb l'anàlisi de seqüències, és a dir, anàlisi de text, so o vídeo. Aquestes, van ser creades a la dècada de 1980. El problema que tenen és que aquestes xarxes han estat molt difícils d'entrenar, ja que els gradients de retropropagació o bé creixen o bé es redueixen amb el pas del temps, de manera que a mesura que passa el temps normalment aquests gradients desapareixen.

Els dos tipus de RNN més coneguts són:

LSTM: Les LSTM resolen el problema de l'entrenament prèviament mencionat problema gràcies al fet que incorporen un seguit de passos per decidir quina informació serà emmagatzemada i quina suprimida. La unitat de memòria LSTM conté tres portes que controlen la manera en què la informació flueix per la unitat.

GRU: Les Gated Recurrent Unit contenen dues portes que controlen la manera en que la informació flueix per la unitat. Aquest tipus de xarxes són més senzilles que les LSTM degut a que contenen menys paràmetres, sense porta de sortida, són més ràpides i la seva execució és més eficient.

Confluint aquest apartat en tenim les GRU que tenen un rendiment més elevant en petits conjunt de dades i les LSTM que proporcionen millors resultats davants de escenaris complexos per treballar amb més quantitat de dades. (13)

2.6. Extracció de característiques d'àudios

Cal depurar la informació a classificar per crear el conjunt de dades d'entrenament. Per això, és necessari fer un pre-processament de cada àudio amb l'objectiu d'extreure la informació més rellevant que permeti resoldre el problema en qüestió, en aquest cas, classificar un conjunt de senyals d'àudio en un determinat nombre de classes. Per això, l'extracció de característiques és una part fonamental dins qualsevol sistema de reconeixement de patrons (14)

Les característiques orientades a àudios, tenen diferent tipus de taxonomies com es mostra a la figura 2.5.

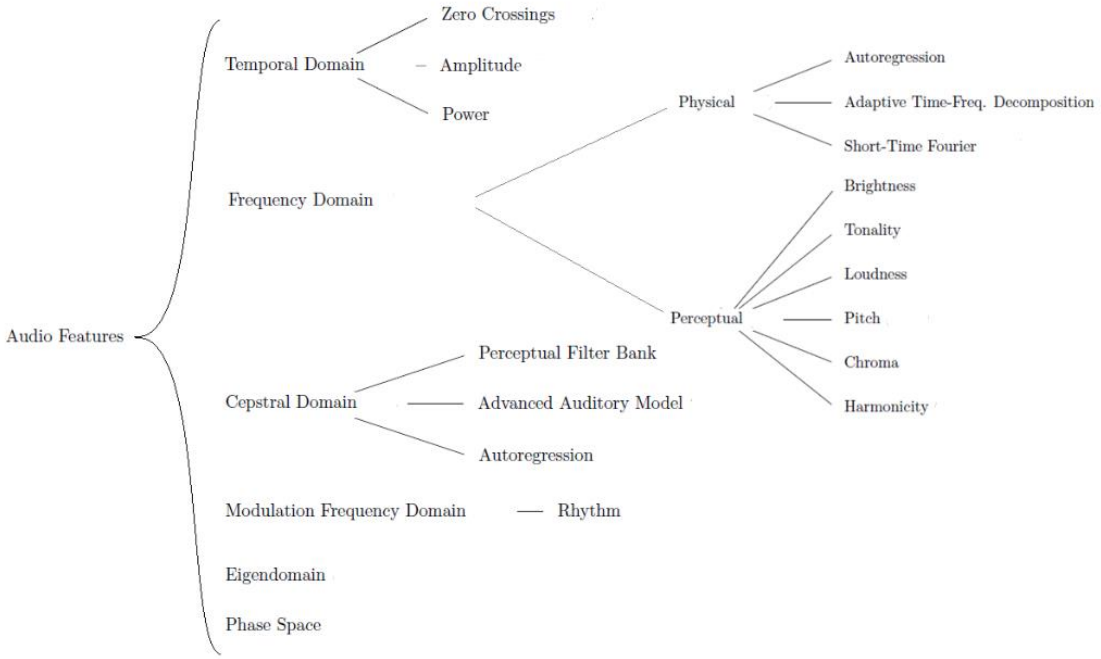


Figura 2.5. Taxonomía de característiques (14).

Classifica les diferents característiques en: temporals, d'energia, espectrals, harmòniques i perceptuals. En aquest treball s'han simplificat la classificació a tres categories:

- **Domini cepstral**

  En el que destaca l'obtenció dels coeficients Mel Cepstral (MFCC, Mel-Frequency Cepstrum Coefficients) ha estat considerada com una de les tècniques de parametrització de la veu més important i utilitzada (Davis and Mermelstein 1980). L'objectiu d'aquesta transformació és obtenir una representació compacta, robusta i apropiada per posteriorment obtenir un model estadístic amb un alt grau de precisió. Un dels inconvenients que tens els paràmetres MFCCs és que el seu mètode de obtenció no està normalitzat, podent haver lleugeres variants entre una i una altra implementació.

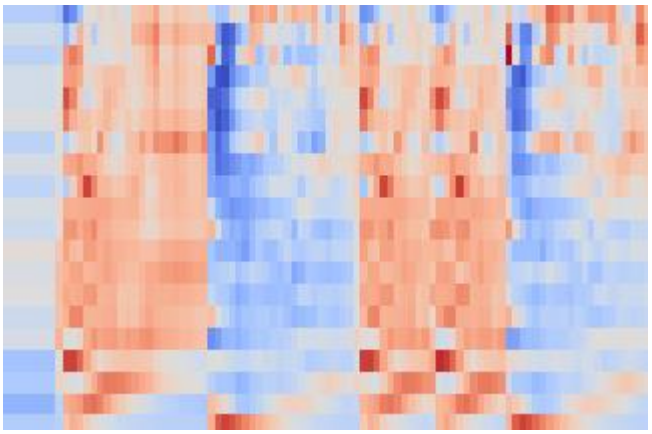


Figura 2.6.MFCC.

- **Domini espectral**

  L'escala Mel relaciona la freqüència percebuda, o to, d'un to pur amb la seva freqüència mesurada real. Els humans són molt millors per discernir petits canvis en el to a freqüències baixes que en freqüències altes. La incorporació d'aquesta escala fa que les nostres característiques coincideixin més de prop amb allò que escolten els humans.

- **Domini temporal**

  Un mètode conegut és el Zero Crossing Rate(ZRC).

  El Zero Crossing Rate és un tipus de característica obtinguda al domini temporal, mesura el nombre de vegades que un senyal creua per zero. El ZCR és útil per determinar si un senyal conté o no veu.

Tenint en compte que en el nostre estudi tots els àudios utilitzats per el training tenen cants d'ocells el ZRC no ens interessa i per la variabilitat del mètode de MFCC per falta de normalització s'ha descartat. A més, en concursos de classificació de àudios com el de BirdCLEF i DCASE els guanyadors han utilitzat els espectrogrames de mel.

Per aquests raonament s'ha escollit la representació de l'àudio es l'escala de mel per el posterior ús en els models.

3. Primer experiment

En aquest capítol s'explica l'algoritme crear per a la classificació de 4 especies de ocells creat en Google Colab, un servei al núvol que proporciona recursos de computació (GPU). Per a començar a treballar primer s'han de aconseguir els àudios dels ocells i penjar-los al compte de Drive associat.

3.1. Eines utilitzades

Per la realització d'aquest experiment i el següent s'ha utilitzat el entorn de programació de Google Colabs en la seva versió gratuïta on s'ha programat en llenguatge Phyton, el drive on s'han emmagatzemat les dades a tractar i els models creats. Com a dispositiu per realitzar les tasques de programació i organització de carpetes s'ha utilitzat un ordinador de sobretaula.

3.2. Obtenció de les dades

Les dades d'entrenament es construeixen a partir de la base de dades de Xeno-Canto (3), una pagina web dedicada a compartir àudios d'ocells de tot el món. Aquest primer experiment conté 201 enregistraments sonors amb un total de 4 espècies. : African Pied Wagtail, Barn Swallow, Black WoodPecker i Black-headed Gull. La majoria dels fitxers d'àudio mostren una gran varietat de qualitat d'enregistrament, durada, recompte d'ocells i soroll de fons. El conjunt d'entrenament té un equilibri de classe . Les dades d'entrenament venen complementades a la pagina web amb una taula d'informació que contenen metadades com ara espècies en primer pla i fons, puntuacions de qualitat dels usuaris i temps i ubicació de l'enregistrament i nom i notes de l'autor. No en vam fer ús de cap de les metadades addicionals, excepte l'identificador de classe de les espècies en primer pla. La presència de nombroses espècies de fons distorsiona les dades d'entrenament i fa una sola etiqueta formació especialment difícil.

Un cop descarregades es van crear una carpeta per el entrenament i una altre per la validació. Dintre de les quals hi ha una carpeta amb el nom de cada espècie d'ocell on es van introduir els àudios 40 àudios de cada espècie van ser utilitzats per el entrenament i 6 àudios per la validació. Un cop realitzar aquesta tasca es penja en el compte del Drive per començar a treballar amb les dades.

3.3. Flux de treball

El nostre flux de treball consta de quatre passos principals. Primer seleccionem un àudio per fer una proba per aconseguir el seu espectrograma. En segon lloc, passem a la obtenció de els espectrogrames de tots els àudios. A continuació intentem trobar una arquitectura CNN que pel fa al nombre de mostres i classes utilitzades aconseguim resultats acceptables. Finalment entrenem el model.

3.4. Generació dels espectrogrames

A partir de els àudios aconseguits es procedeix a transformar-los en imatges per la realització del posterior model de classificació. Aquesta tasca s'ha realitzat mitjançant el llenguatge de programació de Phyton en l'entorn de programació de Google Colabs.

Vaig utilitzar espectrogrames de magnitud amb una resolució de 576x576 píxels, amb una duració de 20 segons que representen la senyal d'àudio. Aquesta mida d'entrada relativament gran és computacionalment car quan s'entrenen ConvNets, però els nostres experiments demostren que els espectrogrames d'alta resolució contenen detalls més valuosos que la classificació general.

Aquest espectrogrames han siguts escalats i aplicats un filtre passa alts per netejar les dades i eliminant interferències de soroll de freqüències que no són d'interès permetre al model extreure les dades del cant d'ocell.

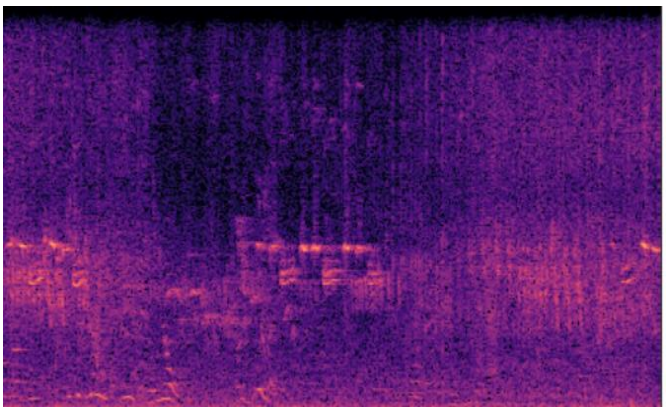


Figura 3.1. Espectograma linear.

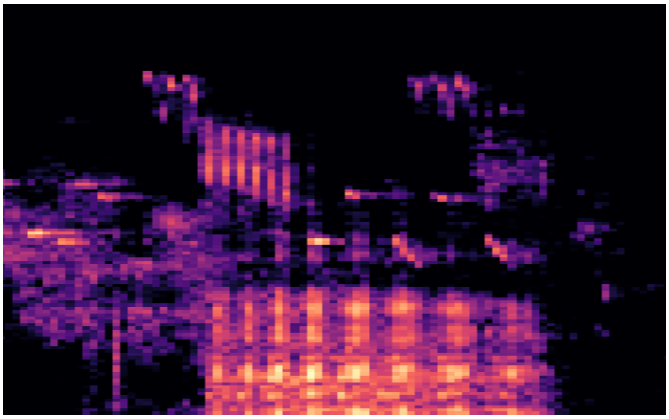


Figura 3.2. Espectograma de mel escalat amb filtre passa alts.

3.5. Arquitectura CNN

Trobar la millor arquitectura de CNN és una tasca que requereix molt de temps i sovint es fa exclusivament per intuïció. D'aquesta manera es va decidir reduir la quantitat de possible disseny i confiar en les millors pràctiques actuals per als dissenys de CNN. Específicament vam utilitzar la EfficientNetB3.

EfficientNet és una arquitectura de xarxa neuronal convolucional i amb un mètode que escala uniformement totes les dimensions de profunditat, amplada, resolució mitjançant un coeficient compost. A diferència de la pràctica convencional que escala arbitràriament aquests factors, el mètode d'escala EfficientNet escala uniformement l'amplada, la profunditat i la resolució de la xarxa amb un conjunt de coeficients d'escala fixos.

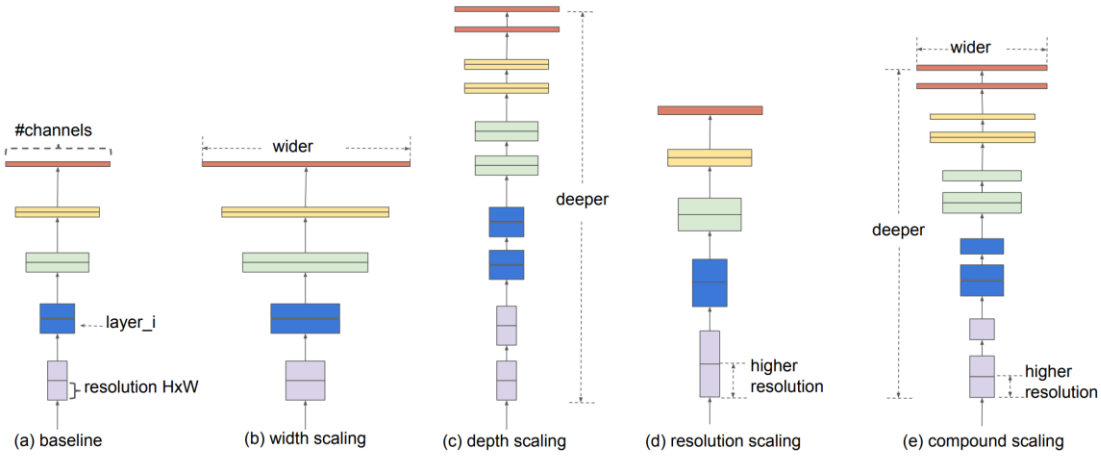


Figura 3.3. Escala del model. La imatge a es la xarxa de referencia. De la b a la d son els escalats convencionals que només incrementen una dimensió de la xarxa. E es la EfficientNet amb un mètode de escalat compost .
(14).

A continuació es mostra la estructura utilitzada per el model de classificació.

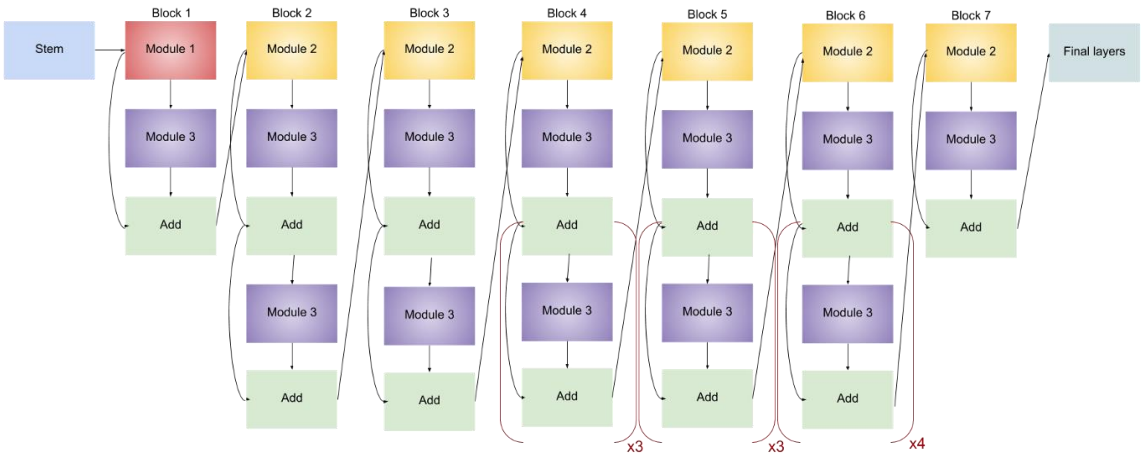


Figura 3.2. Arquitectura EfficientNetB3
(15)(16).

La arquitectura disposa de 11.084.588 paràmetres, dels quals 10.997.292 són entrenables i 87.796 no ho són.

3.6. Entrenament

S'han avaluat diferents tipus de paràmetres:

Taxa d'aprenentatge: la taxa d'aprenentatge és un dels paràmetres més importants en entrenar ConvNets. Les taxes d'aprenentatge fixes poden dificultar l'optimització procés de convergència. Tenint en consideració aquesta característica vam aplicar un callback al programa que fes una reducció d'aquesta característica al moment de trobar-se el model estancat en un resultat de pèrdua del conjunt de validació.

Optimitzador: tria el millor optimitzador per a la actualització del paràmetre de descens del gradient estocàstic és vital per a una convergència d'optimització ràpida. Es va acabar utilitzant del ADAM a causa de l'alta velocitat de convergència que proporciona aquest algoritme.

Pressició: es el paràmetre que es utilitzat per la avaluació del model, s'espera un número relativament elevat per considerar un bon model.

Batch Size: augmentar la mida dels lots per al procés d'entrenament és beneficiós principalment a causa de l'ús de la normalització per lots. Per a Lots més petits condueixen a més iteracions per epoch i tendeixen a tenir un millor rendiment després de les primers epoch. Al final, a més gran els lots semblen oferir una millor generalització. Escollint sempre la millor mida de lot depèn de la quantitat de VRAM que ofereix la GPU. En el nostre cas estem utilitzant els recursos del núvol que ens ofereix l'aplicatiu de Google Colab en la seva versió gratuïta amb això en consideració es va establir la mida del lot a 20.

Epoch : Es realitza l'entrenament del model per 30 epochs, que indica que passa 30 vegades tot el conjunt de dades d'entrenament. Amb els resultats obtinguts es decideix que per els futurs experiments amb 20 epochs es suficient, ja que el loss es comença a estabilitzar sobre aquell número.Com es pot apreciar a la imatge 3.4.

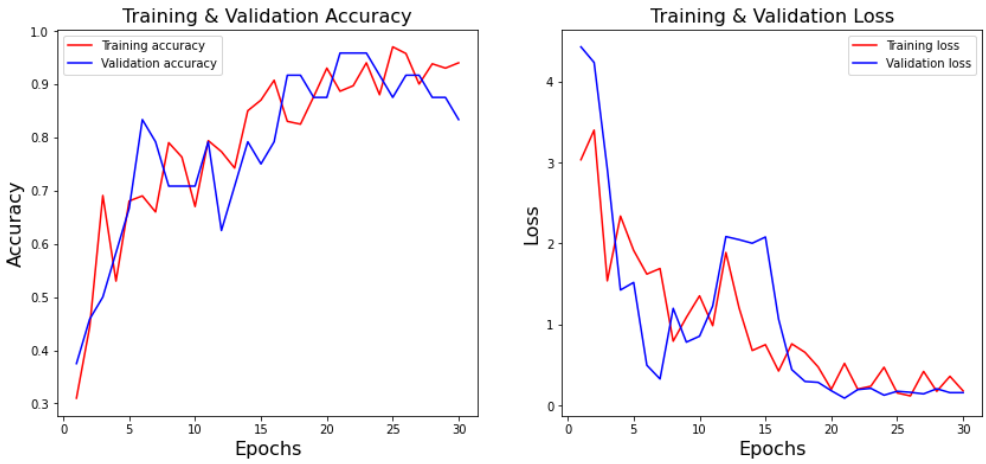


Figura 3.4. Grafiques.Precisió i pèrdua del conjunt de entrenament vermell. Precisió i pèrdua del conjunt de validació blau. Epoch=30.Steps per epoch=5.

3.7. Avaluació

La avaluació d'aquest primer experiment s'ha realitzar separant manualment el conjunt de dades en un conjunt de entrenament i en un altre de validació. On aproximadament el conjunt de validació correspon al 10% del conjunt total de dades. A la figura 3.5 s'aprecien els resultats obtinguts.

```
# Train the model
net_final.fit_generator(train_batches,validation_data=valid_batches,class_weight=class_weight,
    epochs=30,steps_per_epoch=19,callbacks=[ModelCheck, ReduceLR])
19/19 [==============================] - 299s 16s/step - loss: 1.1880 - accuracy: 0.7798 - val_loss: 3.1097 - val_accuracy: 0.7083
Epoch 3/30
19/19 [==============================] - 298s 16s/step - loss: 0.8876 - accuracy: 0.8221 - val_loss: 2.9503 - val_accuracy: 0.7917
Epoch 4/30
19/19 [==============================] - 303s 16s/step - loss: 1.4870 - accuracy: 0.7480 - val_loss: 3.1081 - val_accuracy: 0.6250
Epoch 5/30
19/19 [==============================] - 305s 16s/step - loss: 1.1061 - accuracy: 0.8048 - val_loss: 1.1075 - val_accuracy: 0.8750
Epoch 6/30
19/19 [==============================] - 301s 16s/step - loss: 0.9844 - accuracy: 0.8396 - val_loss: 2.5821 - val_accuracy: 0.5833
Epoch 7/30
19/19 [==============================] - 305s 16s/step - loss: 1.6388 - accuracy: 0.8086 - val_loss: 4.8343 - val_accuracy: 0.3750
Epoch 8/30
19/19 [==============================] - 305s 16s/step - loss: 1.3251 - accuracy: 0.8235 - val_loss: 6.1829 - val_accuracy: 0.7500
Epoch 9/30
19/19 [==============================] - 311s 16s/step - loss: 1.5200 - accuracy: 0.7807 - val_loss: 6.9354 - val_accuracy: 0.7500
Epoch 10/30
19/19 [==============================] - 307s 16s/step - loss: 0.9554 - accuracy: 0.8476 - val_loss: 0.4588 - val_accuracy: 0.9167
Epoch 11/30
19/19 [==============================] - 310s 16s/step - loss: 0.9452 - accuracy: 0.8610 - val_loss: 3.2115 - val_accuracy: 0.7917
Epoch 12/30
19/19 [==============================] - 304s 16s/step - loss: 0.5367 - accuracy: 0.9084 - val_loss: 0.4612 - val_accuracy: 0.9167
Epoch 13/30
19/19 [==============================] - 305s 16s/step - loss: 0.6878 - accuracy: 0.8824 - val_loss: 6.8727 - val_accuracy: 0.5833
Epoch 14/30
19/19 [==============================] - 307s 16s/step - loss: 0.7772 - accuracy: 0.8727 - val_loss: 0.8059 - val_accuracy: 0.9167
Epoch 15/30
19/19 [==============================] - 306s 16s/step - loss: 0.4295 - accuracy: 0.9272 - val_loss: 0.4390 - val_accuracy: 0.8750
Epoch 16/30
19/19 [==============================] - 314s 17s/step - loss: 0.5548 - accuracy: 0.9412 - val_loss: 1.2437 - val_accuracy: 0.8333
Epoch 17/30
19/19 [==============================] - 307s 16s/step - loss: 0.7444 - accuracy: 0.9118 - val_loss: 1.1585 - val_accuracy: 0.8750
Epoch 18/30
19/19 [==============================] - 307s 16s/step - loss: 0.9225 - accuracy: 0.9118 - val_loss: 1.4348 - val_accuracy: 0.8333
Epoch 19/30
19/19 [==============================] - 305s 16s/step - loss: 0.6966 - accuracy: 0.9091 - val_loss: 0.9169 - val_accuracy: 0.8333
Epoch 20/30
19/19 [==============================] - 305s 16s/step - loss: 0.4193 - accuracy: 0.9385 - val_loss: 1.6379 - val_accuracy: 0.7500
Epoch 21/30
19/19 [==============================] - 307s 16s/step - loss: 0.3261 - accuracy: 0.9412 - val_loss: 0.6799 - val_accuracy: 0.9167
Epoch 22/30
19/19 [==============================] - 302s 16s/step - loss: 0.4245 - accuracy: 0.9434 - val_loss: 1.3958 - val_accuracy: 0.9167
Epoch 23/30
19/19 [==============================] - 308s 16s/step - loss: 0.1308 - accuracy: 0.9786 - val_loss: 0.0069 - val_accuracy: 1.0000
Epoch 24/30
19/19 [==============================] - 309s 16s/step - loss: 0.0248 - accuracy: 0.9920 - val_loss: 0.0666 - val_accuracy: 0.9583
Epoch 25/30
19/19 [==============================] - 304s 16s/step - loss: 0.0573 - accuracy: 0.9840 - val_loss: 1.7013 - val_accuracy: 0.9583
Epoch 26/30
19/19 [==============================] - 305s 16s/step - loss: 0.0786 - accuracy: 0.9840 - val_loss: 0.0187 - val_accuracy: 1.0000
Epoch 27/30
19/19 [==============================] - 307s 16s/step - loss: 0.1731 - accuracy: 0.9759 - val_loss: 5.3087e-04 - val_accuracy: 1.0000
Epoch 28/30
19/19 [==============================] - 309s 16s/step - loss: 0.1654 - accuracy: 0.9759 - val_loss: 0.1747 - val_accuracy: 0.9167
Epoch 29/30
19/19 [==============================] - 303s 16s/step - loss: 0.1167 - accuracy: 0.9733 - val_loss: 0.1091 - val_accuracy: 0.9583
Epoch 30/30
19/19 [==============================] - 310s 16s/step - loss: 0.0755 - accuracy: 0.9866 - val_loss: 0.0083 - val_accuracy: 1.0000
<tensorflow.python.keras.callbacks.History at 0x7fbc9c9d8810>
```

Figura 3.5. Resultats de la Efficienet.

L'entrenament de la xarxa neuronal de 4 classes te una durada aproximadament de 3 hores. A partir de la Figura 3.4 de la xarxa neuronal entrenada amb 30 epochs s'observa que l'índex d'error durant el entrenament disminueix progressivament amb el pas del temps, al inici aquesta xarxa neuronal falla molt, però a mida que hi passa el temps, es a dir es van completant els epochs, la xarxa va aprenent augmentant la precisió d'aquesta un 50% en el pas de les primeres 5 epochs. D'aquesta manera l'error s'ha anat reduint de manera progressiva fins a partir del epoch 20 on s'ha mantingut força estable, de fet en aquest moment l'error oscil·la entre valors de 0,5 i 0,1 aproximadament .

El resultat de l'algoritme s'observa a partir de la sortida de la xarxa neuronal com s'aprecia a la figura 3.5 en la sortida apareixen diferents paràmetres d'interès, primerament a l'esquerra l'epoch corresponent a la línia, seguit del temps d'execució el que en un principi no en pot semblar molt ja que aproximadament tots els epochs triguin uns 305 s però al anar augment en número de epoch

que hi volem utilitzar per el conjunt de dades estudiat en pot resultar en un entrenament del model que hi trigui una gran quantitat de hores i fins i tot dies. Per aquest motiu amb la representació gràfica de la pèrdua i la precisió s'extreu el valor òptim d'epochs el qual es tracta d'aquell on la pèrdua es comença a estabilitzar i per tant el model deixa de millorar, reduint el temps empleat per la creació de un model funcional.

Com s'ha vist els valors de precisió del conjunt de validació al final es troben al voltant del 90% aconseguint d'aquesta manera una bona arquitectura per la realització del segon experiment. A més com s'ha comentat prèviament els epochs utilitzats en el següent experiment en seran de 20 ja que volem un bon resultat del model reduint tot el possible el temps de computació sense comprometre la funcionalitat d'aquest.

Al final de la secció del annex es troba el enllaç amb els notebooks utilitzats on poden accedir amb un compte UPC.

4. Segon experiment

En aquest segon capítol s'explica el mètode utilitzar per fer l'estudi de les mostres necessàries per crear un bon model depenent del número de classes que es vol classificar. Les eines utilitzades en aquest experiment en són les mateixes que s'han utilitzat per realitzar el primer experiment, sent aquestes un ordinador de sobretaula i el Google Colabs.

4.1. Obtenció de les dades

La obtenció dels arxius d'àudio es realitza a partir del mateix portal de Xeno-Canto. En aquest cas es classifiquen les dades en subconjunt d'experiments, depenent del número de classes que es pretén estudiar, 2 classes, 3 classes, etc. A més cadascun d'aquest subconjunts en tindran el seu propi conjunt de entrenament, validació y test un conjunt que no s'havia utilitzat en el primer experiment. Aquest conjunt de test en serà l'utilitzat per avaluar la precisió dels models depenent de les classes utilitzades. Els conjunts de validació i test es mantenen constant per tot el subconjunt de classes es a dir en el cas de 2 classes el conjunt de test i validació no canvia malgrat que es vagi augmentant les mostres del conjunt de entrenament. Aquest conjunt d'entrenament esta dividit en diverses carpetes canviant el número de mostres: 10 mostres per classe, 20 mostres per classe, etc. Realitzat aquesta classificació de dades es procedeix a penjar-les al Drive.

4.2. Flux de treball

El nostre flux de treball consta de tres passos principals. Primer, passem a la obtenció de els espectrogrames de tots els àudios. Seguidament s'obté un model òptim per a cada subconjunt de mostres de cada conjunt de classes. Per finalitzar s'avaluen els models obtinguts amb les dades de test, es gràfica i s'observa la relació entre classes, numero de mostres i precisió.

4.3. Generació dels espectrogrames

El procés de generació dels espectrogrames es el mateix que l'utilitzat en el experiment previ.

4.4. Obtenció dels models

S'utilitza la arquitectura de la EfficientNet amb els paràmetres aconseguits en el primer experiment. Per a la obtenció del diferents models es realitza un entrenament de l'arquitectura amb diferents conjunts de d'entrenament en el que varia la quantitat de mostres utilitzades i el conjunt de validació corresponent al numero de classes estudiades en aquell moment. Llavors per l'obtenció del millor model es fan servir callbacks, en concret un que et guarda el millor model a partir del valor del loss que aconsegueix en el conjunt de validació. Aquests models son desats en el drive per utilitzar-los mes endavant.

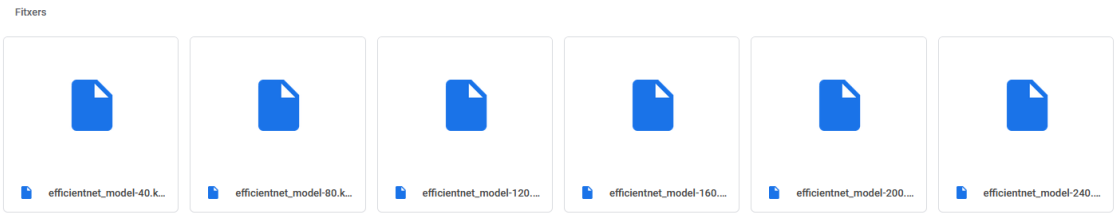


Figura 4.1. Models desats en format .keras per als diferents subconjunt de mostres en el cas d'estudi de 4 classes de la Efficienet.

4.5. Avaluació dels models

En aquest últim pas del experiment consisteix en carregar al notebook de Google Colabs els models desats en el drive i avaluar-los amb la utilització del conjunt de dades que en pertany al test. El resultat d'aquesta operació es la precisió del model carregat sobre el conjunt de dades de test. Per una avaluació d'aquest resultats es va procedir a graficar la precisió obtinguda en relació del número de mostres utilitzats. A continuació en les figures 4.2, 4.3, 4.4 i 4.5 es mostren les corbes d'aprenentatges obtingudes.

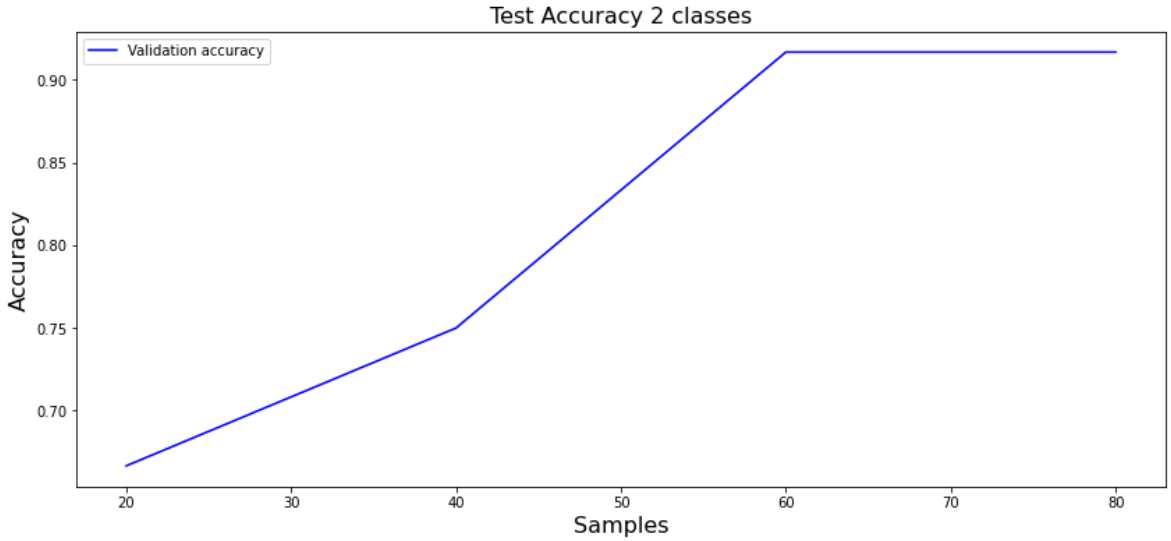


Figura 4.2. Corba d'aprenentatge per l'estudi de 2 classes.

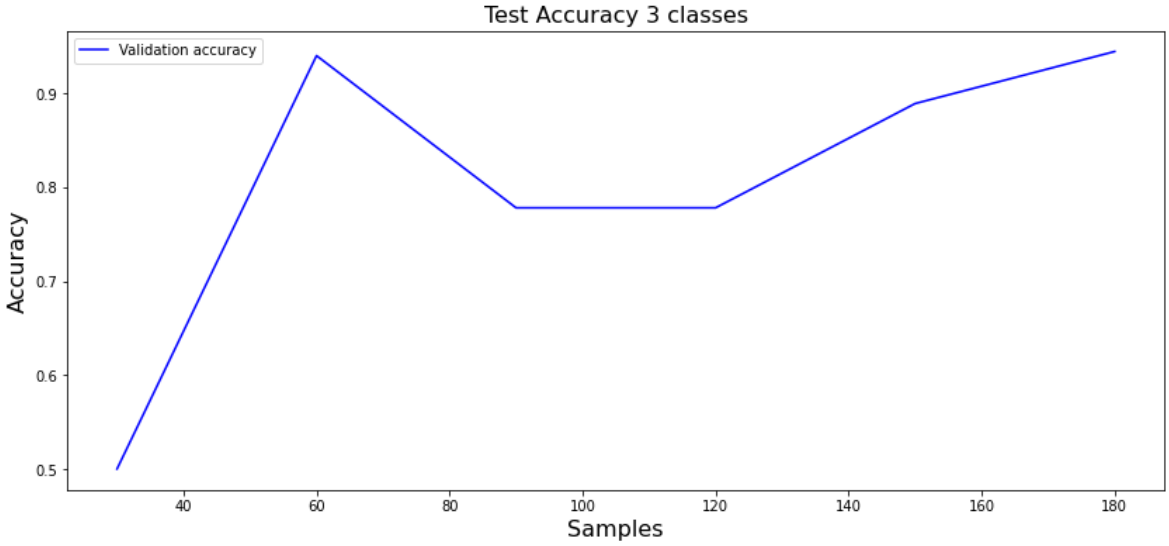


Figura 4.3. Corba d'aprenentatge per l'estudi de 3 classes.

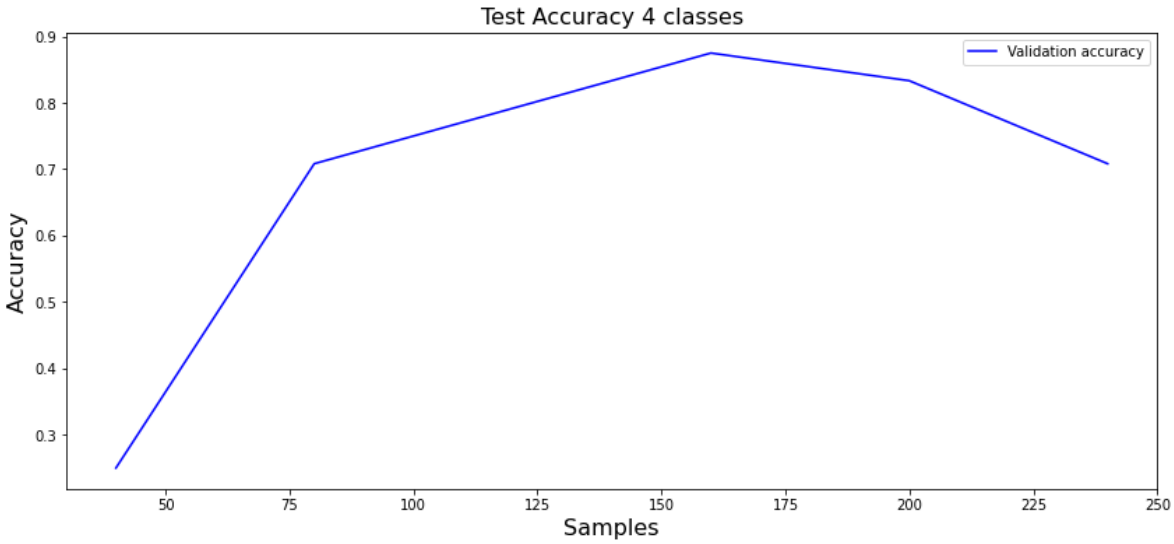


Figura 4.4. Corba d'aprenentatge per l'estudi de 4 classes.

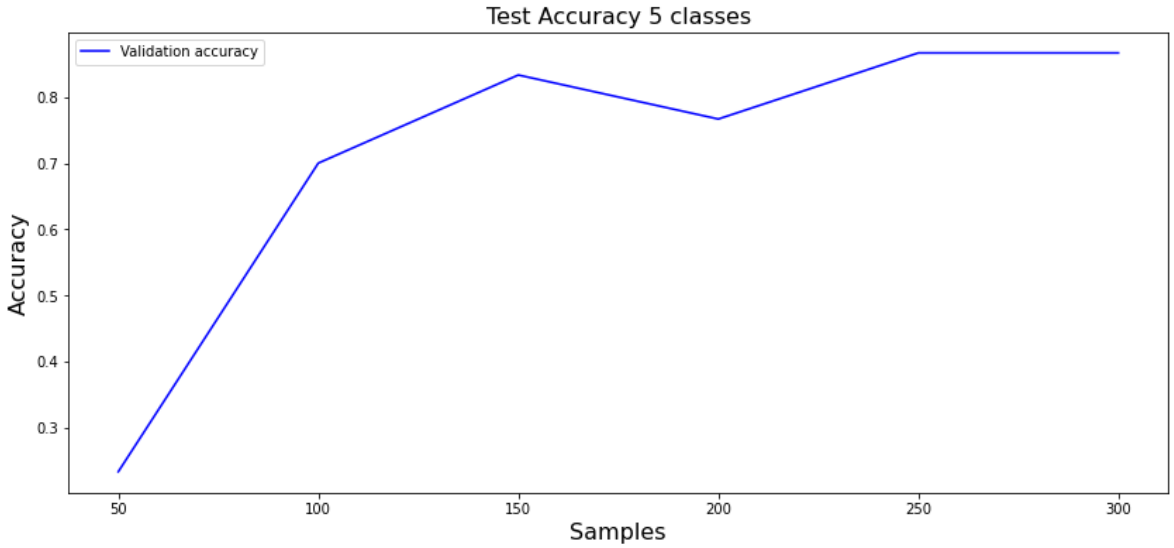


Figura 4.5. Corba d'aprenentatge per l'estudi de 4 classes.

S'examina l'efecte de la mida de la mostra sobre la precisió del resultat de la classificació. La mida de la mostra es va manipular i va oscil·lar depenent del número de classes. Per a 2 classes la mostra oscil·la entre 20 i 80, per 3 entre 30 i 180, per 4 entre 40 i 240 i per 5 entre 50 i 300 mostres.

En les gràfiques obtingudes es pot apreciar la tendència general de la millora en la precisió del models a mida que s'augmenten les classes malgrat que en alguns casos com en el de la figura 4.3 hi existeix un pic en la mostra 60 que no es representatiu, el qual es pot deure a que justament el model creat amb les 60 mostres es ideal per les mostres utilitzades com el set de test. A més, la quantitat requerida de mostres per la obtenció d'un bon model augmentat només en un el número de classes estudiat es gran con s'observa comparant l'estudi de 2 classes amb la de 3, que en el cas de les 2 classes es requereixen només 2 mostres mentre que en el cas de les 3 classes es necessiten unes 180 mostres.

En la figura 4.4 es veu un decreixement del valor de precisió al final de la corba, però comparant amb els resultats de la figura 4.5 en la qual hi ha una baixada de precisió entre 150 i 200 mostres es probable que si augmentéssim el número de mostres del experiment de 4 classes tornés a fer una lleugera pujada. Per els cassos de 2 i 5 classes s'ha aconseguit estabilitzar el valor de precisió pont dir així que per les 2 classes es necessiten 60 mostres mentre que per les 5 en necessitaríem 250.

Al final de la secció del annex es troba el enllaç amb els notebooks utilitzats on poden accedir amb un compte UPC.

5. Anàlisi de l'impacte ambiental

En aquest estudi l'impacte mediambiental es dedueix a partir de les emissions de CO2 a l'atmosfera com a resultat del consum d'energia elèctrica per la utilització de l'ordinador de sobretaula. Tenint en compte que el consum mitjà d'electricitat del dispositiu es de 760 kWh i prenent com a relació entre CO2 i kWh es de 1 kWh emet uns 450 grams de CO2 s'ha realitzat una emissió de 342 kg de CO2 a l'atmosfera.

Conclusions i Treball Futur

S'ha proporcionat informació sobre el nostre intent de classificació del so dels ocells a petita escala utilitzant xarxes neuronals convolucionals.

El primer problema que es va afrontar va ser trobar una base de dades sobre la que treballar ja que es necessita un mínim de mostres per que els algoritmes posteriorment aplicats siguin funcionals, després de fer una primera investigació en el portal de Kaggel no es van trobar dades suficients per començar els experiments, però sí que en una publicació del portal sortia una pàgina d'on havien extret les dades dels ocells trobant així la pàgina de xeno-canto un repositori de base de dades d'ocells de fanàtics d'aquell món.

Seguidament s'ha decidit com s'hauria de fer l'apropament al problema, prenent els resultats del guanyadors de concursos de classificació d'àudios d'ocells es va implementar el pre-processament de les dades d'àudio transformant-les en espectrogrames de mel de forma exitosa en un primer àudio individual. Posteriorment es va fer la adaptació de la funció que generava l'espectrograma per crear un bucle que en crees múltiples a partir de fer un recorregut dintre de una carpeta on s'havien desat els àudios. Aconseguit aquesta meta es va procedir a implementar un primer model per classificar 4 especies d'ocells, el model escollit va ser un tipus de CNN ja que al igual que els espectrogrames de mel era una eina recurrent en la resolució d'aquest problemes en els concursos prèviament esmentats. Es van identificar els paràmetres més importants en la implementació de la xarxa com pot ser l'optimitzador i el número de epochs, aconseguint així un primer model per a les 4 classes que en fa una bona classificació dels àudios d'ocells. Tota aquesta feina es la base per aconseguir un model sobre el que treballar en el segon experiment.

A continuació, amb l'arquitectura obtinguda del primer experiment es van crear diversos models per diferents número de mostres per obtenir el desenvolupament de la precisió a partir de les mostres utilitzades, al arribar a 6 classes es va trobar que no es podia realitzar el estudi ja que la base de dades de xeno-canto per algunes classes no hi havia més mostres sobre les que treballar, impedint veure el desenvolupament per a classes superiors a 5. S'hauria llavors de aconseguir més mostres per altres vies per poder realitzar un estudi amb més profunditat. Malgrat això, per conjunts amb poques classes s'han obtingut les mides necessàries per la creació de bons models amb el numero de classes estudiats.

Per altre banda, en el desenvolupament del treball s'han observat punts de millora i de treball futur.

Com a millora del treball realitzat seria la utilització de el Google Colabs Pro, ja que l'eina gratuïta te limitacions de capacitat computacional impedint realitzar algunes instruccions de cop per consum de RAM, a més això permetria una major velocitat en el temps de compilació dels diversos notebooks .

Treball futur

Introduint els possibles treballs futurs podem començar amb augmentar el número de classes de l'estudi de les mostres necessàries.

Per altre banda en el món real en tenim el problema que molts cops els àudios de ocells estan barrejats i ens podria interessar detectar no només la classe predominant sinó totes les classes que hi apareixen en aquell àudio, llavors en lloc de fer un problema monolabel, d'una capa s'hauria de realitzar un multilabel, es a dir multicapa.

L'experiment en qüestió consistiria en mirar si amb els models obtinguts es poden distingir arxius multietiqueta sense necessitat d'entrenar-los. Per aconseguir-lo s'hauria de canviar la funció d'activació de la capa de sortida que en el nostre problema d'una única etiqueta es tracta de softmax a una neurona de sigmoid. Aquest experiment no s'ha realitzat en aquest treball per falta de temps.

Finalment, es considera un treball exitós, malgrat les millores aplicables amb la utilització del software premium o en defecte d'un ordinador amb millor capacitat computacional.

Anàlisi Econòmica

Aquest projecte va consistir principalment en investigació i desenvolupament, per això s'ajusta el pressupost aquestes dues tasques, pel que fa a llicències i eines utilitzades per al desenvolupament, els recursos eren Codi obert i llicències comunitàries gratuïtes.

Els costos resultants basats en les explicacions anteriors són generalment el preu mínim per hora d'estudiant en pràctiques fixada per l'EEBE, en 8€/h per a la temporada universitària 2021/2022 i el consum electric s'ha calculat a partir de una calculadora online(17) tal i com es veu a la figura 5.

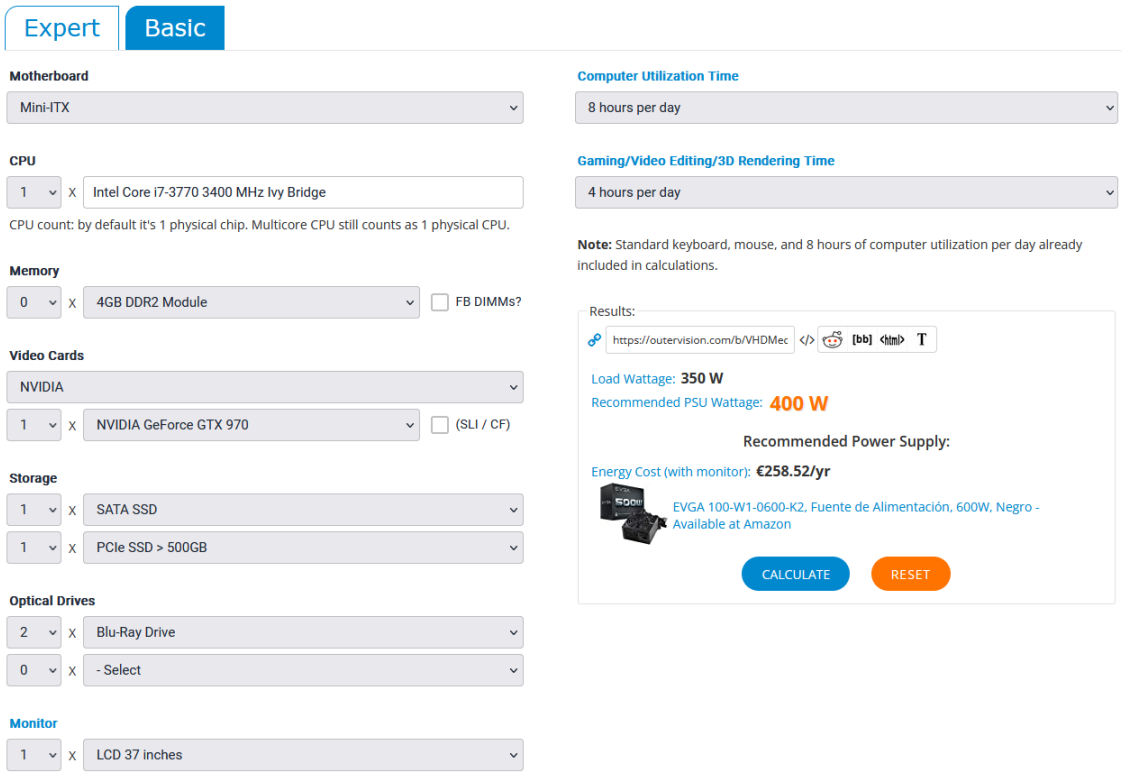


Figura 5.Càlcul consum elèctric.

Així, el pressupost d'aquest projecte és:

Hores	Preu per any (€/any)	Preu per hores (€/h)	Total(€)
320	258,52	-	86,17
500	-	8	4000
			4086,17€

Taula 1.Càlcul consum elèctric.

Bibliografia

1. "Bird audio detection, Challenge results", Dcase, accedit Febrer 2022,

https://dcase.community/challenge2018/task-bird-audio-detection-results#Lasseck_MfN

2. Magdalena Kortas, "SOUND-BASED BIRD CLASSIFICATION",Towards Data Science, accedit Febrer 2022,

https://towardsdatascience.com/sound-based-bird-classification-965d0ecacb2b

3. "Sharing bird sounds from around the world",Xeno-canto, accedit Març 2022,

https://www.xeno-canto.org

4. Prashant Sharma, "Applications of Convolutional Neural Networks(CNN)",Analytics Vidhya, publicat 4 Octubre 2021,

https://www.analyticsvidhya.com/blog/2021/10/applications-of-convolutional-neural-networkscnn/

5. "Welcome to Colaboratory",Google, accedit Març 2022,

https://colab.research.google.com/notebooks/intro.ipynb

6. "Will a robot take your job?", BBC, accedit Març 2022,

https://www.bbc.com/news/technology-34066941

7. "Deep Learning",IBM,accedit Març 2022,

https://www.ibm.com/cloud/learn/deep-learning

8. Ayush Parhi, "Deep Learning: The State of the art", The Startup, accedit Març 2022,

https://medium.com/swlh/deep-learning-the-state-of-the-art-88153e69a602

9. Yann LeCun, Yoshua Bengio i Geoffrey Hinton, "Deep learning",*Nature*, Vol. 521(28 Maig 2015): 436-444,

https://www.cs.toronto.edu/~hinton/absps/NatureDeepReview.pdf

10. “Machine Learning y Deep Learning. Todo lo que necesitas saber”, Spartanhack, última modificació 11 de Desembre,2020,

https://spartanhack.com/machine-learning-y-deep-learning-todo-lo-que-necesitas-saber/

11. Sumit Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way”,Towards Data Science, accedit Març 2022,

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

12. Margalith,”Convolutional Neural Network”, DataScientest, última modificació 16 de Desembre, 2021,

https://datascientest.com/es/convolutional-neural-network-es

13. Diego Calvo,” Red Neuronal Recurrente – RNN”, Diegocalvo, última modificació 9 de Desembre, 2018,

https://www.diegocalvo.es/red-neuronal-recurrente/

14. Dalibor Mitrovic, Matthias Zeppelzauer i Christian Breiteneder,” Features for Content-Based Audio Retrieval”, *Advances in Computers* Vol. 78 (2010): 71-150,

https://www.researchgate.net/publication/220662548_Features_for_Content-Based_Audio_Retrieval

15. Mingxing Tan i Quoc V. Le,” EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, 28 Maig de 2019,

https://arxiv.org/abs/1905.11946

16. Vardan Agarwal, “Complete Architectural Details of all EfficientNet Models”, Towards Data Science, Abril 2022,

https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b736142

17. “Power Supply Calculator”, Extreme Outer Vision, Juny 2022,

https://outervision.com/power-supply-calculator

18. " pathlib — Filesystem Paths as Objects" , PyMOTW, última modificació 18 de Març de 2018,

https://pymotw.com/3/pathlib/

19. "pathlib — Object-oriented filesystem paths",Phyton, accedit Març 2022,

https://docs.python.org/3/library/pathlib.html

20. "tf.keras.preprocessing.image.ImageDataGenerator", TensorFlow, última modificació 28 de Maig de 2022,

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

21. "tf.keras.applications.efficientnet.EfficientNetB3",TensorFlow, última modificació 18 de Maig de 2022,

https://www.tensorflow.org/api_docs/python/tf/keras/applications/efficientnet/EfficientNetB3

22. Shipra Saxena," Introduction to Softmax for Neural Network", Analytics Vidhya, publicat 5 Abril de 2021,

https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/

23. "tf.keras.Model", TensorFlow, última modificació 3 Juny 2022,

https://www.tensorflow.org/api_docs/python/tf/keras/Model

24. "tf.keras.callbacks.ReduceLROnPlateau", TensorFlow, última modificació 28 Maig 2022,

https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ReduceLROnPlateau

25. "librosa.feature.melspectrogram",Librosa, , accedit Març 2022,

https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html

Annex A

En aquest annex s'explica el codi implementat en els diferents notebooks.

El conjunt de l'entrenament serà l'utilitzat per la creació del model i el conjunt de la validació per comprovar la precisió del model creat.

Al Google Colab s'ha de muntar e Google Drive per tenir accés a les dades que s'han pujat. A través de la comanda com s'observa a la Figura A.1.

```
from google.colab import drive
drive.mount("/content/drive")
```

Figura A.1. Accés del Colab al Drive

Un cop muntat el Drive cal fer uns canvis de versions de biblioteques concretament de l'hp5y,tensorflow,keras i segmentation-models.

```
!pip install 'h5py==2.10.0'
!pip install tensorflow==2.0
!pip install keras==2.3.1
!pip install segmentation-models
```

Figura A.2. Instalació de versions de biblioteques

El conjunt de llibreries que seran utilitzades per que es pugui desenvolupar correctament el codi es mostra en la figura A.3.

```
import pandas as pd
import efficientnet.tfkeras as efn
import IPython.display as ipd
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from efficientnet.keras import preprocess_input
from keras.callbacks.callbacks import ModelCheckpoint, ReduceLROnPlateau
from sklearn.utils import class_weight
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import librosa
import skimage.io
import pathlib
import os
import keras
from keras.models import Sequential
import warnings
```

Figura A.3. Conjunt de llibreries.

Per a la creació d'un model de classificació el primer pas que hem de fer es transformar les dades en forma de àudio a dades tipus imatge. Per aconseguir-lo el que es realitza es la creació d'espectrogrames de mel i la seva posterior desada en el drive per treballar amb ells en passos posteriors.

Aquesta idea ha sigut extreta dels guanyadors de diferents concursos de classificar ocells, els que van compartir el seu apropament al problema com el de BirdCLEF i DCASE.

Per a la creació des espectrogrames s'utilitza la llibreria librosa(25) , aquesta es troba dintre de diferents for, que serveix per fer un recorregut entre tots els arxius que es troben dintre de cadascuna de les carpetes especificades gracies al ús de la llibreria pathlib que permet introduir la ruta dels arxius (18) (19).

Creant la seva imatge corresponent i guardant-lo en la adreça especificada al final, linea que utilitza el nom del arxiu original de àudio i canvia la seva extensió a png. Tal i com s'aprecia en la Figura A.3.

```
plt.figure(figsize=(8,8))

species = 'African Pied Wagtail', 'Barn Swallow', 'Black Woodpecker', 'Black-headed Gull'
for s in species:
    pathlib.Path(f'/content/drive/My Drive/models/joan/img_data_train/{s}').mkdir(parents=True, exist_ok=True)
    for filename in os.listdir(f'/content/drive/My Drive/data/intel-artf/tfg/xeno ocells/songs_clean_extended_train/{s}'):
        songname = f'/content/drive/My Drive/data/intel-artf/tfg/xeno ocells/songs_clean_extended_train/{s}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=20)
        N_FFT = 1024
        HOP_SIZE = 1024
        N_MELS = 128
        WIN_SIZE = 1024
        WINDOW_TYPE = "hann"
        FEATURE = "mel"
        FMIN = 1400
        mel = librosa.feature.melspectrogram(y=y,sr=sr,n_fft=N_FFT,hop_length=HOP_SIZE,n_mels=N_MELS,htk=True,fmin=FMIN,fmax=sr / 2,)
        librosa.display.specshow(librosa.power_to_db(mel ** 2, ref=np.max), fmin=FMIN, y_axis="linear")
        plt.colorbar(format="%+2.0f dB")
        plt.axis('off');

        plt.savefig(f'/content/drive/My Drive/models/joan/img_data_train/{s}/{filename[:-4].replace(".", "")}.png')
        plt.clf()
```

Figura A.4. Obtenció dels espectrogrames.

Un com realitzat això per el conjunt de entrenament i de validació ja en tenim les dades preparades per crear el model de classificació.

Amb el conjunt de dades creades ara s'han de carregar al programa, per fer aquesta tasca definim les direccions on es troben les imatges per el entrenament i per a la validació. A més també hi creem una variable en forma de llista on hi trobem les 4 possibles classes dels ocells.

```
species = ['African Pied Wagtail', 'Barn Swallow', 'Black Woodpecker', 'Black-headed Gull']
path_train = '/content/drive/My Drive/models/joan/img_data_train/'
path_validation = '/content/drive/My Drive/models/joan/img_data_validation/'
```

Figura A.5. Direcció i especies de ocells.

A la imatge A.6 es troba el codi utilitzat per a carregar les imatges en el algoritme i fer un pre-processament. S'agafa una fracció del total de la amplada i alçada de la imatge, ja que width_shift_range i el height_shift_range tenen assignats valors menors a 1. A la part de flow_from directory es en el lloc on definim d'on extreu les imatges, les classes correspon a la llista que s'ha definit abans amb les especies que hi existeixen. En el cas del conjunt de entrenament definim shuffle com True perquè ens barregi les mostres del conjunt de entrenament (20). Per finalitzar es defineix el numero de processos abans de que el model s'actualitzi en la variable Btch, que en son 20.

```
Btch=20
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.1,
    fill_mode="nearest",
)
train_batches = train_datagen.flow_from_directory(
    path_train,
    classes=species,
    target_size=(224, 224),
    class_mode="categorical",
    shuffle=True,
    batch_size=Btch,
)

valid_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
valid_batches = valid_datagen.flow_from_directory(
    path_validation,
    classes=species,
    target_size=(224, 224),
    class_mode="categorical",
    shuffle=False,
    batch_size=Btch,
)
```

Figura A.6. Carrega i pre-processament d'imatges.

Amb això el programa troba 177 imatges que corresponen al conjunt de entrenament i 24 pel conjunt de validació, per la realització del primer experiment. Tenint una proporció de aproximadament 90% de conjunt de entrenament i 10% de conjunt de validació.

Creació del model

El model creat no en té una capa completament connectada a la sortida, definida posant el paràmetre de include_top a False. S'han utilitzar pesos pre-entrenats del conjunt de dades ImageNet, definit en el codi com weights="imagenet". Al definir el primer paràmetre esmentat com a Flase s'ha de definir input_shape que es una dimensió obligada a ser formada per 3 canals (21).

La variable x es la responsable d'emmagatzemar la sortida de la nostra efficientnet, que la guarda com un vector (4,1) en el nostre cas ja que estem tractant amb 4 classes. Apliquem un flatten per transformar-lo a un vector (1,4). Per evitar el overfitting en el nostre model s'aplica un Dropout.

La quantitat de classes del model queda definida tenint en compte la longitud del vector de les especies definit anteriorment (22).

Al tractar-se d'un problema superior a 2 classes s'aplica el procés de predicció matemàtica softmax, la qual calcula la probabilitat relativa de pertànyer a les especies.

Per acabar compilem el model utilitzant l'optimitzador de Adam que consisteix en mètode de descens del gradient estocàstic que es basa en l'estimació adaptativa de moments de primer i segon ordre. Categorical_crossentropy és una mètrica que crea dues variables locals, total i count que s'utilitzen per calcular la freqüència amb què y_pred coincideix amb y_true. Tenint com a mètrica de avaluació de model la precisió (23). La xarxa neuronal s'observa en la figura A.7.

```
net = efn.EfficientNetB3(include_top=False, weights="imagenet", input_tensor=None, input_shape=(224, 224, 3))
x = net.output
x = Flatten()(x)
x = Dropout(0.5)(x)
output_layer = Dense(len(species), activation="softmax", name="softmax")(x)
net_final = Model(inputs=net.input, outputs=output_layer)
net_final.compile(optimizer=Adam(), loss="categorical_crossentropy", metrics=["accuracy"])
```

Figura A.7. Característiques del model.

El següent pas a realitzar abans de començar a entrenar el model es definir uns callbacks (24).

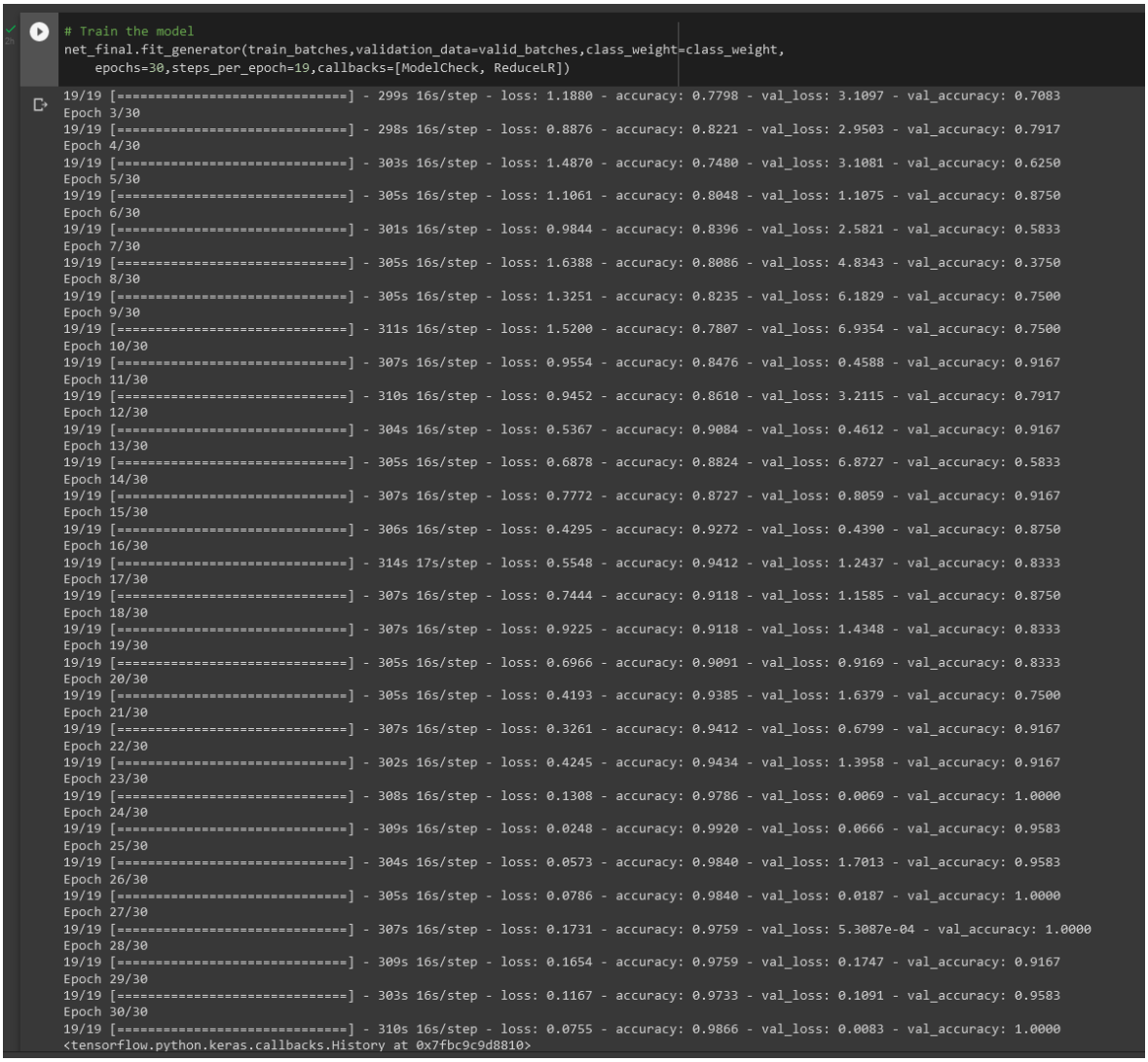
El ModelCheck serveix perquè desi el millor model creat en el moment que faci el entrenament basat en el millor valor del loss del conjunt de dades de la validació. Mentre que el ReduceLR redueix la velocitat d'aprenentatge en el moment que la mètrica que esta avaluant, el loss de la validació en el nostre cas deixa de millorar. La implementació es representada a la figura A.8.

```
ModelCheck = ModelCheckpoint(
    "/content/drive/My Drive/models/efficientnet_checkpoint.h5",
    monitor="val_loss",
    verbose=0,
    save_best_only=True,
    save_weights_only=True,
    mode="auto",
    period=1,
)

ReduceLR = ReduceLROnPlateau(monitor="val_loss", factor=0.2, patience=5, min_lr=3e-4)
```

Figura A.8. Callbacks.

Per finalitzar es realitza l'entrenament del model per 30 epochs, indica que passa 30 vegades tot el conjunt de dades d'entrenament. Els resultat s'observen a la figura 3.5.



```
# Train the model
net_final.fit_generator(train_batches,validation_data=valid_batches,class_weight=class_weight,
    epochs=30,steps_per_epoch=19,callbacks=[ModelCheck, ReduceLR])
```

```
19/19 [==============================] - 299s 16s/step - loss: 1.1880 - accuracy: 0.7798 - val_loss: 3.1097 - val_accuracy: 0.7083
Epoch 3/30
19/19 [==============================] - 298s 16s/step - loss: 0.8876 - accuracy: 0.8221 - val_loss: 2.9503 - val_accuracy: 0.7917
Epoch 4/30
19/19 [==============================] - 303s 16s/step - loss: 1.4870 - accuracy: 0.7480 - val_loss: 3.1081 - val_accuracy: 0.6250
Epoch 5/30
19/19 [==============================] - 305s 16s/step - loss: 1.1061 - accuracy: 0.8048 - val_loss: 1.1075 - val_accuracy: 0.8750
Epoch 6/30
19/19 [==============================] - 301s 16s/step - loss: 0.9844 - accuracy: 0.8396 - val_loss: 2.5821 - val_accuracy: 0.5833
Epoch 7/30
19/19 [==============================] - 305s 16s/step - loss: 1.6388 - accuracy: 0.8086 - val_loss: 4.8343 - val_accuracy: 0.3750
Epoch 8/30
19/19 [==============================] - 305s 16s/step - loss: 1.3251 - accuracy: 0.8235 - val_loss: 6.1829 - val_accuracy: 0.7500
Epoch 9/30
19/19 [==============================] - 311s 16s/step - loss: 1.5200 - accuracy: 0.7807 - val_loss: 6.9354 - val_accuracy: 0.7500
Epoch 10/30
19/19 [==============================] - 307s 16s/step - loss: 0.9554 - accuracy: 0.8476 - val_loss: 0.4588 - val_accuracy: 0.9167
Epoch 11/30
19/19 [==============================] - 310s 16s/step - loss: 0.9452 - accuracy: 0.8610 - val_loss: 3.2115 - val_accuracy: 0.7917
Epoch 12/30
19/19 [==============================] - 304s 16s/step - loss: 0.5367 - accuracy: 0.9084 - val_loss: 0.4612 - val_accuracy: 0.9167
Epoch 13/30
19/19 [==============================] - 305s 16s/step - loss: 0.6878 - accuracy: 0.8824 - val_loss: 6.8727 - val_accuracy: 0.5833
Epoch 14/30
19/19 [==============================] - 307s 16s/step - loss: 0.7772 - accuracy: 0.8727 - val_loss: 0.8059 - val_accuracy: 0.9167
Epoch 15/30
19/19 [==============================] - 306s 16s/step - loss: 0.4295 - accuracy: 0.9272 - val_loss: 0.4390 - val_accuracy: 0.8750
Epoch 16/30
19/19 [==============================] - 314s 17s/step - loss: 0.5548 - accuracy: 0.9412 - val_loss: 1.2437 - val_accuracy: 0.8333
Epoch 17/30
19/19 [==============================] - 307s 16s/step - loss: 0.7444 - accuracy: 0.9118 - val_loss: 1.1585 - val_accuracy: 0.8750
Epoch 18/30
19/19 [==============================] - 307s 16s/step - loss: 0.9225 - accuracy: 0.9118 - val_loss: 1.4348 - val_accuracy: 0.8333
Epoch 19/30
19/19 [==============================] - 305s 16s/step - loss: 0.6966 - accuracy: 0.9091 - val_loss: 0.9169 - val_accuracy: 0.8333
Epoch 20/30
19/19 [==============================] - 305s 16s/step - loss: 0.4193 - accuracy: 0.9385 - val_loss: 1.6379 - val_accuracy: 0.7500
Epoch 21/30
19/19 [==============================] - 307s 16s/step - loss: 0.3261 - accuracy: 0.9412 - val_loss: 0.6799 - val_accuracy: 0.9167
Epoch 22/30
19/19 [==============================] - 302s 16s/step - loss: 0.4245 - accuracy: 0.9434 - val_loss: 1.3958 - val_accuracy: 0.9167
Epoch 23/30
19/19 [==============================] - 308s 16s/step - loss: 0.1308 - accuracy: 0.9786 - val_loss: 0.0069 - val_accuracy: 1.0000
Epoch 24/30
19/19 [==============================] - 309s 16s/step - loss: 0.0248 - accuracy: 0.9920 - val_loss: 0.0666 - val_accuracy: 0.9583
Epoch 25/30
19/19 [==============================] - 304s 16s/step - loss: 0.0573 - accuracy: 0.9840 - val_loss: 1.7013 - val_accuracy: 0.9583
Epoch 26/30
19/19 [==============================] - 305s 16s/step - loss: 0.0786 - accuracy: 0.9840 - val_loss: 0.0187 - val_accuracy: 1.0000
Epoch 27/30
19/19 [==============================] - 307s 16s/step - loss: 0.1731 - accuracy: 0.9759 - val_loss: 5.3087e-04 - val_accuracy: 1.0000
Epoch 28/30
19/19 [==============================] - 309s 16s/step - loss: 0.1654 - accuracy: 0.9759 - val_loss: 0.1747 - val_accuracy: 0.9167
Epoch 29/30
19/19 [==============================] - 303s 16s/step - loss: 0.1167 - accuracy: 0.9733 - val_loss: 0.1091 - val_accuracy: 0.9583
Epoch 30/30
19/19 [==============================] - 310s 16s/step - loss: 0.0755 - accuracy: 0.9866 - val_loss: 0.0083 - val_accuracy: 1.0000
<tensorflow.python.keras.callbacks.History at 0x7fbc9c9d8810>
```

Figura 3.5. Resultats de la efficienet.

Els resultats obtinguts es situen per sobre del 90%.

Gràfics

Per fer una posterior valoració del número de epochs necessaris per l'obtenció d'un bon model es realitzen 2 gàfiques. El codi es troba en la figura 3.3.1. Mentre que les gàfiques a la figura A.9.

```
import matplotlib.pyplot as plt

history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
accuracy = history_dict['accuracy']
val_accuracy = history_dict['val_accuracy']

epochs = range(1, len(loss_values) + 1)
fig, ax = plt.subplots(1, 2, figsize=(14, 6))
#
# Plot the model accuracy vs Epochs
#
ax[0].plot(epochs, accuracy, 'r', label='Training accuracy')
ax[0].plot(epochs, val_accuracy, 'b', label='Validation accuracy')
ax[0].set_title('Training & Validation Accuracy', fontsize=16)
ax[0].set_xlabel('Epochs', fontsize=16)
ax[0].set_ylabel('Accuracy', fontsize=16)
ax[0].legend()
#
# Plot the loss vs Epochs
#
ax[1].plot(epochs, loss_values, 'r', label='Training loss')
ax[1].plot(epochs, val_loss_values, 'b', label='Validation loss')
ax[1].set_title('Training & Validation Loss', fontsize=16)
ax[1].set_xlabel('Epochs', fontsize=16)
ax[1].set_ylabel('Loss', fontsize=16)
ax[1].legend()
```

Figura A.9. Codi per graficar.

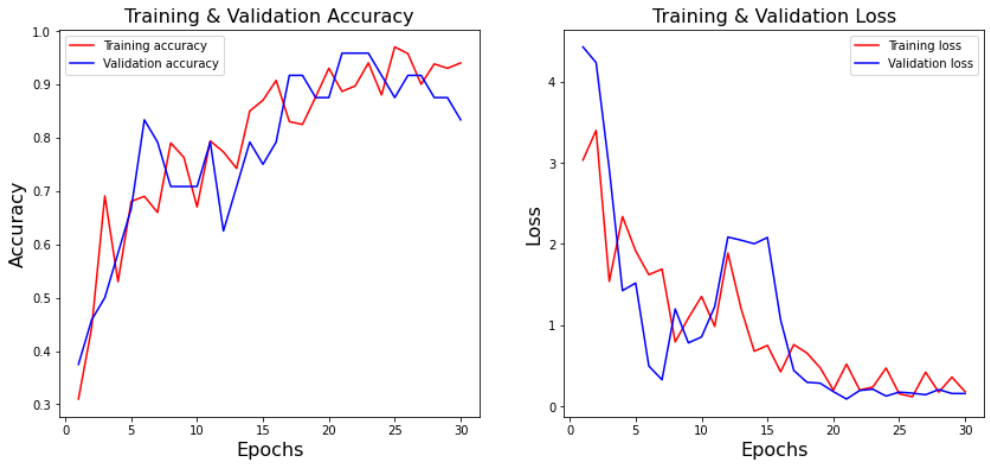


Figura 3.4. Grafiques.Precisió i pèrdua del conjunt de entrenament vermell. Precisió i pèrdua del conjunt de validació blau. Epoch=30.Steps per epoch=5.

Podem valorar com un bon número de epochs aquell que aconsegueix una pèrdua propera a 0 i el valor de pèrdua deixa de disminuir.

Avaluació amb conjunt de test

L'algoritme crear per a la representació de les diferents corbes d'aprenentatge de diferents grups d'espècies utilitza la metodologia prèviament explicada per la creació dels espectrogrames i la creació del model.

A diferencia del primer apartat es crea un nou grup de dades anomenat test, el qual utilitzarem per avaluar els diferents models de classificació que s'han anat desat amb el ModelCheck.

A continuació es carreguen els models i s'avaluen amb el conjunt de test, tal i com s'observa ala figura A.10.

```
model = tf.keras.models.load_model("/content/drive/My Drive/models/5 classes/efficientnet_model-50.keras")
result1=model.evaluate(test_batches)
print(result1)
```

```
2/2 [==============================] - 12s 6s/step - loss: 3.5473 - accuracy: 0.2333
[3.547260284423828, 0.23333333]
```

Figura A.10. Avaluació del model de 5 classes de 50 mostres.

I al iterar la funció per els diferent models es procedeix a representar gràficament els resultats obtinguts. Representat en la figura A.11.

```
test_accuracy=[0.2333,0.7000,0.8333, 0.7667,0.8667,0.8667]
samples_total=[50,100,150,200,250,300]

fig, ax = plt.subplots(1,figsize=(14, 6))
#
# Plot the test accuracy vs Samples
#

ax.plot(samples_total, test_accuracy, 'b', label='Validation accuracy')
ax.set_title('Test Accuracy 5 classes', fontsize=16)
ax.set_xlabel('Samples', fontsize=16)
ax.set_ylabel('Accuracy', fontsize=16)
ax.legend()
```

Figura A.11. Gráfica de corba d'aprenentatge de 5 classes.

Enllaç

Els notebooks es poden accedir a partir del següent enllaç:

https://drive.google.com/drive/folders/1KESDErIGOldymWy3Ap63oekJQVRs7hsI?usp=sharing