Classificació d'emocions facials amb Xarxa Neuronal Convolucional

Albert Ramos Nadal i Pau Villén Vidal
Universitat Politècnica de Catalunya

(Dated: December 2024)

1. Introducció

Aquest projecte es basa en el desenvolupament d'un model basat en xarxes neuronals convolucionals (CNN), amb l'obejctiu de classificar imatges en escala de grios en funció de les emocions facials.
Per tal de realitzar-ho, s'ha utilitzat un dataset que conté imatges separades en sis emocions diferents: "happy", "angry", "surprise", "sad, "neutral" i "fear". Aquest dataset ha estat descarregat a partir de la pàgina web de Kaggle [1].
A partir d'aquestes imatges, s'ha seguit una implementació basada en preprocessament de les imatges, construcció i disseny del model, entrenament, validació, i avaluació a partir de diverses mètriques i resultats visuals.
Per al disseny de la xarxa neuronal, s'ha partit d'idees de diversos codis públics de ([1]), que s'han anat modificant i adaptant per al nostre cas per tal d'aconseguir els millors resultats possibles.
Finalment, també es detallen els resultats obtinguts, el procés d'obtenció d'aquests resultats, les limitacions del projecte i possibles millores futures.

2. Experiments

2.1. Models i eines

Per a la realització d'aquest projecte, s'ha utilitzat el llenguatge de programació Python (fent ús d'Anaconda, Spyder). A partir d'aquest, s'han utilitzat diverses biblioteques i llibreries. Entre aquestes es troben: *NumPy* per a operacions bàsiques, *Seaborn* per a visualització de resultats, *Matplotlib* també per a visualització de resultats en forma de gràfiques i, sobretot, TensorFlow per a la construcció de la xarxa CNN.
L'arquitectura del model, que serà explicada amb més detall en la següent secció, s'ha basat en una xarxa neuronal convolucional amb diverses capes convolucionals per a extreure informació de les imatges, d'agrupació i densament connectades. També s'ha afegit regularització a aquestes capes per tal de reduir el sobreajustament o *overfitting*, que consisteix en què el model aprèn massa bé els detalls del conjunt d'entrenament i el seu rendiment en dades noves no és tan bo. Finalment, el ritme d'aprenentage utilitzat per a l'optimització del model ha sigut de 0.0001, que també serà comentat més tard.
Cal comentar que el nombre d'execucions ha sigut elevat a causa del disseny d'aquesta arquitectura del model. Es va començar a partir d'un model més bàsic i es van anar afegint capes i canviant diversos paràmetres (especificats posteriorment) fins a aconseguir els resultats desitjats i un bon funcionament de la xarxa neuronal. Addicionalment, també hi ha hagut alguna execució més per tal d'aconseguir la millor visualització possible dels resultats.

2.2. Descripció de les dades

Les dades d'aquest projecte consisteixen en les imatges provinents del dataset descarregat. Aquestes estan dividides en dues carpetes (*train* i *test*) corresponents al conjunt d'imatges per a l'entrenament i la validació del model, respectivament. Totes les imatges estan en escala de grisos i tenen una mida de 48x48.
Originalment, el dataset descarregat tenia 28709 imatges per al conjunt de *train* i 3589 per al conjunt de test, així com una categoria més ("disgust") a part de les esmentades anteriorment. Per al nostre projecte, aquesta última categoria ha estat eliminada, ja que contenia moltes menys imatges que la resta. A més, també s'ha reduït el conjunt d'imatges de la majoria dels conjunts, de manera que es redueix el temps d'execució i s'assegura que totes les categories continguin el mateix nombre de dades, que es mostrarà en la següent secció.
Abans de la construcció del model, hi ha diverses funcions, també explicades posteriorment, per al preprocessament i la visualitzaió de les imatges utilitzades.
A continuació, les dades es processen amb els generadors d'imatges del model aplicant normalització i permetent rotacions i desplaçaments, entre d'altres, per a una millor generalització del model.

2.3. Hardware utilizat

Per acabar, el Hardware utilitzat ha estat una CPU MacBook Air amb xip Apple M1 i RAM 8GB.

3. Implementació

En aquesta secció es detalla la implementació del codi per parts així com els resultats que s'han obtingut en cada pas d'aquesta implementació.
El model funciona a través de l'extracció de característiques de les imatges mitjançant capes convolucionals, que identifiquen patrons visuals que després són analitzats per capes densament connectades per a la classificació en la predicció final. En aquest procés el model aprèn a associar cada imatge amb la seva emoció corresponent.
En primer lloc es descriu el funcionament de cada part del codi i després s'introdueixen els resultats que s'han anat obtenint durant la realització del projecte.

3.1. Funcions prèvies a la construcció del model

En primer lloc, el codi comença important totes les llibreries necessàries, esmentades en la secció anterior. A continuació, s'especifiquen els directoris corresponents per als dos conjunts d'imatges, *train* i *test*.
Un cop els directoris estan definits, com s'ha comentat anteriorment, tenen lloc diverses funcions per tractar el conjunt d'imatges utilitzat.
La primera d'aquestes compta el nombre d'imatges que conté cadascuna de les categories i retorna un *data frame* (Figura 1), que mostra aquests nombres d'imatges.

```
        happy   sad  fear  surprise  neutral  angry
TRAIN:   2999  2999  2999      2999     2999   2999
       happy  sad  fear  surprise  neutral  angry
TEST:    599  599   599       599      599    599
All images from 'train' are already 48x48 pixels.
All images from 'test' are already 48x48 pixels.
```

Figure 1. Mostra per pantalla de les dues funcions prèvies al model. Data frame amb el nombre d'imatges per categoria i missatge informant que les imatges ja tenien la mida desitjada.

La següent funció comprova la mida de les imatges, i, en cas que alguna no tingui la mida desitjada de 48x48 píxels, la redimensiona. Per a acabar, retorna per pantalla una frase que mostra si s'ha hagut de redimensionar alguna imatge o no (Figura 1).
Per acabar, un bucle recórre totes les emocions del directori de *train* i selecciona una imatge aleatòriament, per tal de poder-ne visualitzar una de cada categoria, com es veu a continuació:

Figure 2. Visualització d'una imatge de cada categoria.

3.2. Construcció del model

Al llarg de la realització d'aquest projecte, l'arquitectura del model ha anat variant de paràmetres, capes i esctructura fins a obtenir els millors resultats. En aquesta secció es descriu com s'ha anat construint i millorant el model, així com els resultats que s'han obtingut en cadascun dels casos.

3.2.1. Generadors d'imatges

Abans de la construcció del model, es preparen i creen els generadors d'imatges.
Aquests generadors permeten aplicar transformacions aleatòries a les imatges, de manera que hi hagi més variabilitat en les imatges sense necessitar augmentar-ne el nombre. En el nostre cas, s'apliquen rotacions, translacions, zoom i inversions al generador de les imatges de *train*, aconseguint així un posterior millor entrenament del model.

Addicionalment, s'aplica una normalització als dos subconjunts d'imatges que reescala els valors dels píxels entre 0 i 1, permetent una millor convergència del model.
Un cop preparats els dos generadors, es creen especificant el directory corresponent, la mida de les imatges i el batch size, que fa que es proporcionin les imatges en lots, reduint la memòria necessitada.
Finalment, també es basen en un flux continu, generant les dades de manera continuada i en temps real, sense haver-les de carregar totes a la vegada, optimitzant així l'entrenament.
A la següent figura (3) es veu com s'han definit aquests generadors d'imatges, amb els paràmetres que s'han usat finalment després d'haver estat modificats fins a ser òptims:

```
#Preparem els generadors d'imatges per al model. Introduim algunes transformacions
#aleatòries a les imatges per millorar la generalització del model
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,  #Normalització dels píxels
    rotation_range=30,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1.0 / 255.0)


#Creem els generadors d'imatges anteriorment preparats. Especifiquem el directori
#d'on provenen, la mida desitjada, el batch size i la categoria corresponent.
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode="categorical",
    shuffle = True
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode="categorical",
    shuffle=False
)
```

Figure 3. Codi corresponent als generadors d'imatges per al model.

3.2.2. Arquitectura del model

La xarxa neuronal convolucional consta de diversos tipus de capes que, combinades, permeten l'aprenentatge del model.
Per començar les capes convolucional són les que permeten extreure característiques locals de les imatges, com poden ser vores i patrons. Aquestes capes apliquen filtres convolucionals capaços de detectar aquestes característiques. Cadascuna d'aquestes capes convolucionals s'aplica especificant el nombre de filtres a utilitzar (32, 64, 128, ...). Amb un nombre de filtres més petits es detecten les característiques més senzilles i amb un nombre més gran característiques més complexes i abstractes.
Per aquest motiu, a mesura que ha anat avançant el projecte hem anat afegint capes amb més filtres fins a obtenir millors resultats.
A cadascuna de les capes convolucionals se li aplica una regularització L2. Aquesta penalitza els pesos grans del model, reduint el risc de sobreajustament i millorant la generalització.
En segon lloc, després de cada dues capes convolucionals, s'aplica una capa de normalització (*Batch Normalization*), una capa *MaxPooling* i una capa *Dropout*.
El *Batch Normalization* normalitza els valors d'activació, reduint la variabilitat interna i accelerant la convergència durant l'entrenament. El *MaxPooling* selecciona només els valors més destacats d'una regió, permetent així reduir el nombre de paràmetres per evitar l'*overfitting*. Finalment, el *Dropout* descativa de manera aleatòria una fracció de les neurones durant l'entrenament, evitant excessiva dependència de certs camins neuronals.
Per acabar amb l'arquitectura del model, s'afegeixen capes densament connectades, que són les encarregades de combinar les característiques extretes per les capes convolucionals i fer-ne una classificació segons les 6 emocions.
En l'última capa s'hi aplica Activació *Softmax*, que consisteix en transformar les sortides finals en probabilitats normalitzades per a cada categoria, facilitant així la posterior predicció.
En la següent figura (4) es veu aquesta arquitectura del model per al codi finalment usat:

```
#Capes convolucionals per a la creació de la xarxa
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 1),kernel_initializer="glorot_uniform", padding='same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same',kernel_regularizer=l2(0.01)))
model.add(Conv2D(256, (3, 3),activation='relu', padding='same', kernel_regularizer=l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same',kernel_regularizer=l2(0.01)))
model.add(Conv2D(1024, (3, 3),activation='relu', padding='same', kernel_regularizer=l2(0.01)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

#Capes denses
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Figure 4. Codi corresponent a l'arquitectura del model.

3.2.3. Compilació del model

Per a la compilació del model, s'utilitza l'optimitzador Adam, que fa robust el model. Addicionalment, es defineix el ritme d'aprenentage com a 0.0001. Aquest paràmetre determina la velocitat amb què el model ajusta els seus pesos. El valor utilitzat indica que hi ha una convergència suau i ajustaments petits.
En aquesta part també es defineixen la funció de pèrdua i la mètrica de precisió o *accuracy*, que s'utilitzen per avaluar els resultats i seran explicats posteriorment.
A continuació, la funció *summary* retorna una taula amb un resum de les capes i els paràmetres del model.
Les següents línies de codi (Figura 5) corresponen a aquesta compilació del model:

```
#Compilació del model
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary() #Retorna un resum de les capes i paràmetres del model
```

Figure 5. Codi corresponent a la compilació del model.

3.2.4. Entrenament del model

Per acabar amb aquesta secció, l'última part consisteix en l'entrenament del model. Per a aquest, els principals hiperparàmetres utilitzats són el nombre d'èpoques i el *Batch size*.
El nombre d'èpoques consisteix en quantes vegades el model veu totes les dades d'entranament. Per a aquest projecte s'han utilitzat 20 èpoques, ja que són suficients per assolir una bona convergència i s'ha vist que per molt que s'augmentessin els resultats ja no milloraven més.
Per altra banda, el *Batch size* defineix quantes imatges s'envien alhora al model durant l'entrenament. És a dir, defineix la mida de cada bloc d'imatges que s'envia al model. El valor escollit és de 64, que redueix la memòria necessària per a aquest procés.
EL nombre de passos per època durant l'entrenament està relacionat amb el *Batch size*, és a dir, correspon al nombre total d'imatges dividit per 64, resultant en 281 passos per època.
Finalment, s'introdueix també un criteri de parada anticipada (*Early Stopping* que atura l'entrenament en cas que la funció de pèrdua no millori durant tres èpoques i es queda amb els pesos més òptims d'entre totes les èpoques.
La següent figura (6) mostra el codi per a aquesta part:

```
#Detectar si a partir d'algun moment deixa d'haver-hi millores. Es queda amb els
#pesos més òptims d'entre totes les èpoques
earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True)

callbacks = [earlystop]


#Entrenament del model
history = model.fit(
    train_dataset,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=epochs,
    validation_data=test_generator,
    validation_steps=test_generator.samples // batch_size,
    callbacks=callbacks
)
```

Figure 6. Codi corresponent a l'entrenament del model.

Per acabar, cada cop que s'executa el codi, es mostra per pantalla com va avançant l'entrenament del model pas per pas i època per època, com es mostra en la següent imatge (7).

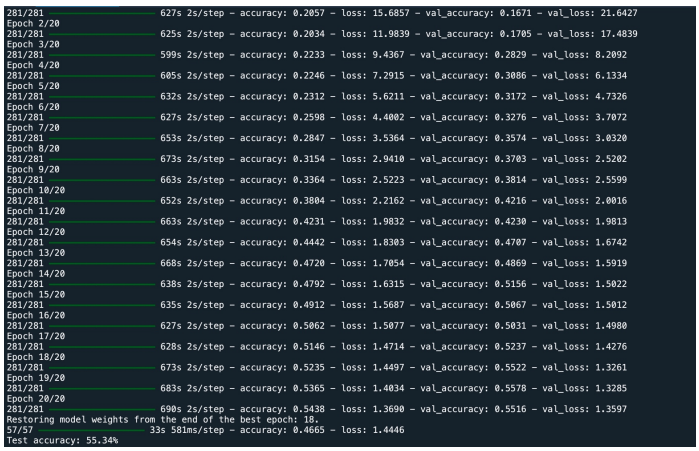


Figure 7. Mostra per pantalla de les èpoques del model en una execució.

3.3. Anàlisi de resultats

Un cop tota la implementació relacionada amb la construcció del model ha estat explicada, en aquesta secció es descriu quins resultats s'obtenen i quines maneres d'analitzar-los s'utilitzen.
Per tal d'avaluar el rendiment del model s'utilitzen la mètrica de precisió o *accuracy* i la funció de pèrdua. La primera consisteix en la proporció de prediccions correctes respecte el total, es pot expressar en percentatge i s'avalua com es veu en la figura 8.

```
#Avaluació de la precisió del model
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test accuracy: {test_accuracy * 100:.2f}%")
```

Figure 8. Codi corresponent a l'avaluació del model.

La funció de pèrdua o *loss* representa el valor de la funció que el model intenta minimitzar durant l'entrenament, indicant la capacitat del model de predir imatges no vistes. Calcula la discrepància entre les probabilitats predites pel model i les categories reals, per tant, valors més baixos representen millor convergència del model.
A continuació, es converteixen les probabilitats extretes pel model en prediccions per al conjunt de *test*.
Un cop el model ha estat avaluat, s'utilitzen tres maneres diferents d'analitzar els resultats, que són explicades a continuació. Tots aquests analisis de resultats seran entesos més fàcilment en la següent secció, on es mostren els resultats obtinguts.
Addicionalment, a part d'analitzar el resultats, el model fa una predicció sobre 20 imatges del conjunt de *test* per avaluar el seu rendiment.

3.3.1. Gràfics

La primera manera més senzilla d'analitzar els resultats consisteix en graficar la precisió i pèrdua prèviament avaluades en funció de les èpoques, tant per al *train* com per al *test*. Es mostra com la precisó va augmentant i la pèrdua va disminuint fins als seus valors finals.

3.3.2. Matriu de confusió

En segon lloc, s'analitzen els resultats mitjancçant la matriu de confusió o *confusion matrix*. Aquesta mostra quines categories són més encertades i quines més confoses entre elles.
Per a aconseguir-ho, la gràfica resultant té totes les categories tant en l'eix x com en l'eix y. L'eix x correspon a la categoria o emoció predita pel model i l'eix y a la categoria correcta. Els números que apareixen en la matriu representen el nombre d'imatges que s'han predit com a aquella certa categoria (eix x) quan la seva etiqueta correcta era la corresponent a l'eix y. Per tant, el resultat òptim seria que els nombres més elevats es situessin a la diagonal de la matriu, indicant que cada emoció ha estat predita més vegades correctament. Això també representa que la diagonal s'ha de veure com més fosca millor, ja que colors més clars indiquen nombres més petits i colors més foscos nombres més grans.
Aquesta matriu de confusió, com el seu nom indica, permet també descobrir quines són les categories més confoses pel model entre elles.
Per obtenir tant la matriu de confusió com el posterior informe de classificació s'utilitza la llibreria *sklearn*.

3.3.3. Informe de classificació

Per acabar, l'ultima manera d'analitzar els resultats és extreure un informe de classificació o *classification report*. Aquest proporciona tres mètriques differents: precisió o *precision*, sensibilitat o *recall* i *f1-score*.
La precisió representa la proporció de prediccions correctes sobre el total de realitzades. La sensibilitat és la proporció d'elements classificats respecte al total de la seva categoria i, finalment, el *f1-score* consisteix en una mitjana harmònica entre *precision* i *recall*.
Cadascuna de les propietats es mostren per a cada categoria, per a una mitjana aritmètica entre elles i per a una ponderada. De totes maneres, la mitjana ponderada no és rellevant en aquest cas ja que hi ha el mateix nombre d'imatges per a cada emoció.
La variable *support* mostra el nombre total d'imatges en cada cas.

4. Resultats obtinguts

Un cop tota la implementació ha estat descrita detalladament, aquesta secció mostra i comenta els resultats que s'han obtingut per diverses versions del codi utilitzat.

4.1. Model senzill

En començar el projecte, es va utilitzar en primer lloc una implementació més bàsica, en què l'arquitectura del model era menys complexa del que ha resultat al final.
Per a aquesta primera implementació s'aplicaven només dues capes convolucionals, amb 32 i 128 filtres, respectivament. A més, encara no s'havien tingut en compte els regularitzadors l2 i s'aplicaven menys vegades les capes *Dropout*. L'arquitectura d'aquest primer model es pot observar en la següent imatge (9), que correspon al resum mostrat per pantalla:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 32)	320
batch_normalization (BatchNormalization)	(None, 48, 48, 32)	128
max_pooling2d (MaxPooling2D)	(None, 24, 24, 32)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	36,992
batch_normalization_1 (BatchNormalization)	(None, 24, 24, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 128)	0
flatten (Flatten)	(None, 18432)	0
dense (Dense)	(None, 256)	4,718,848
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 6)	1,542

Total params: 4,758,342 (18.15 MB)
Trainable params: 4,758,022 (18.15 MB)
Non-trainable params: 320 (1.25 KB)

Figure 9. Resum mostrat per pantalla de l'arquitectura del primer model implementat.

Els resultats obtinguts per aquest model es poden veure en els gràfics següents (10), obtenint una precisió en test de 47.22%, indicant que pot ser encara millorat.

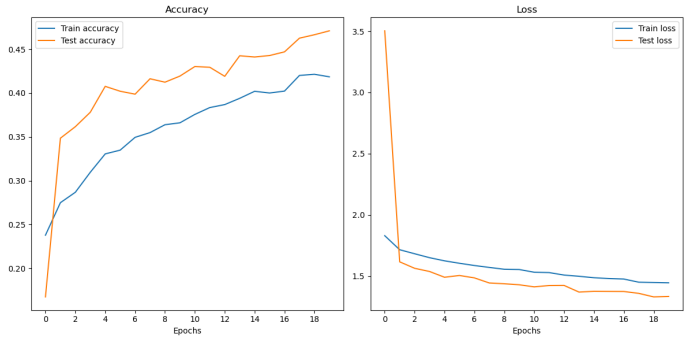
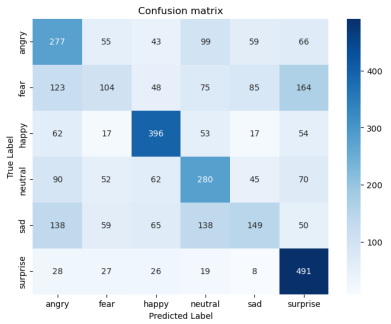


Figure 10. Gràfics per la precisió i pèrdua per al primer model implementat.

Finalment, s'analitzen també els resultats usant la *confusion matrix* i el *classification report* (11). Es pot observar que, encara que algunes categories com "happy" o "surprise" obtenen bons resultats, d'altres com "fear" o "sad" obtenen pitjors resultats. A causa d'aquestes dues categories, la visualització de la matriu de confusió no es correspon amb l'esperat, ja que la diagonal no es mostra de color blau fosc en tots els casos. Addicionalment es mostra també en el classifiation report aquesta diferència entre categories, sobretot per a "fear", que obté valors molt baixos de precisió i sensibilitat.



(a) Matriu de confusió

```
Classification Report
=====================
              precision    recall  f1-score   support

       angry       0.39      0.46      0.42       599
        fear       0.33      0.17      0.23       599
       happy       0.62      0.66      0.64       599
     neutral       0.42      0.47      0.44       599
         sad       0.41      0.25      0.31       599
    surprise       0.55      0.82      0.66       599

    accuracy                           0.47      3594
   macro avg       0.45      0.47      0.45      3594
weighted avg       0.45      0.47      0.45      3594
```

(b) Informe de classificació

Figure 11. Matriu de confusió i informe de classificació per a la primera implementació del model.

4.2. Model final

Partint de les conclusions obtingudes en els resultats del primer model implementat, es va anar afegint capes i modificant paràmetres del model durant la realització del procés. El model ha tingut diferents versions durant aquesta, no obstant, expliquem a continuació la implementació final del model amb la que s'han obtingut els millors resultats.

Els paràmetres i capes utilitzats per a aquesta versió final del model són els que s'han descrit a la secció anterior de Implementació. Respecte al model prèviament descrit, s'han canviat paràmetres dels generadors d'imatges, s'ha disminuit el ritme d'aprenentatge fins a 0.0001, s'ha afegit el criteri de parada anticipada (no usat abans) i, sobretot, s'han afegit més capes.

En primer lloc, les capes convolucionals han augmentat, afegint-ne de fins a 1024 filtres. A més, s'han afegit els regularitzadors l2 prèviament mencionats i s'han modificat les capes densament connectades i les *Dropout*, com es veu a continuació en la taula resum:

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 48, 48, 32)	320
conv2d_13 (Conv2D)	(None, 48, 48, 64)	18,496
batch_normalization_6 (BatchNormalization)	(None, 48, 48, 64)	256
max_pooling2d_6 (MaxPooling2D)	(None, 24, 24, 64)	0
dropout_8 (Dropout)	(None, 24, 24, 64)	0
conv2d_14 (Conv2D)	(None, 24, 24, 128)	73,856
conv2d_15 (Conv2D)	(None, 24, 24, 256)	295,168
batch_normalization_7 (BatchNormalization)	(None, 24, 24, 256)	1,024
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 256)	0
dropout_9 (Dropout)	(None, 12, 12, 256)	0
conv2d_16 (Conv2D)	(None, 12, 12, 512)	1,180,160
conv2d_17 (Conv2D)	(None, 12, 12, 1024)	4,719,616
batch_normalization_8 (BatchNormalization)	(None, 12, 12, 1024)	4,096
max_pooling2d_8 (MaxPooling2D)	(None, 6, 6, 1024)	0
dropout_10 (Dropout)	(None, 6, 6, 1024)	0
flatten_2 (Flatten)	(None, 36864)	0
dense_4 (Dense)	(None, 1024)	37,749,760
dropout_11 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 6)	6,150

Figure 12. Resum mostrat per pantalla de l'arquitectura del model final implementat.

Els resultats obtinguts per aquest model es poden veure en els gràfics següents (13), obtenint una precisió en test de 55.34%, indicant que ha millorat notòriament respecte al primer model.

Es pot veure també que la curves de les gràfiques tenen un comportament molt més continu que anteriorment, indicant que el model segueix un aprenentatge més progressiu i amb menys fluctuacions.

No obstant, el fet que la precisió per a l'entrenament i el *test* siguin semblants en les últimes èpoques indica que s'està aprop del sobreajustament. Per aquest motiu s'intenta implementar una millora en la següent secció.

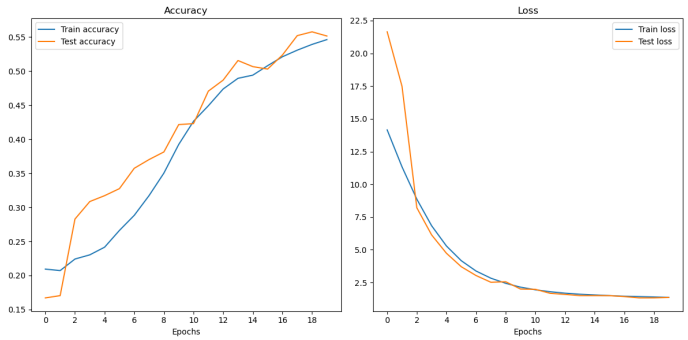
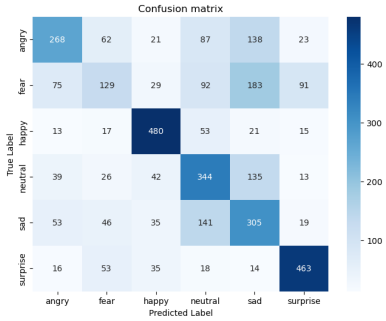


Figure 13. Gràfics per la precisió i pèrdua per al model final implementat.

Com s'ha fet anteriorment, la *confusion matrix* i el *classification report* són representats (14) per a aquest cas.



(a) Matriu de confusió

Classification Report

	precision	recall	f1-score	support
angry	0.58	0.45	0.50	599
fear	0.39	0.22	0.28	599
happy	0.75	0.80	0.77	599
neutral	0.47	0.57	0.52	599
sad	0.38	0.51	0.44	599
surprise	0.74	0.77	0.76	599
accuracy			0.55	3594
macro avg	0.55	0.55	0.54	3594
weighted avg	0.55	0.55	0.54	3594

(b) Informe de classificació

Figure 14. Matriu de confusió i informe de classificació per a la implementació final model.

En aquest cas, els resultats han millorat respecte l'anterior implementació. La matriu de confusió mostra ja més clarament la diagonal fosca esperada, amb nombres més elevats, i l'informe de classificació mostra valors més elevats tant per a precisió com sensibilitat.
De totes maneres, el fet que algunes de les emocions obtinguin molt pitjors resultats que d'altres segueix suposant un problema per al rendiment del model, que representa la principal limitació per a aquest projecte.
Es pot veure en l'informe de classificació que les categories "sad", "fear" i "angry" tenen un resultat molt pitjor en comparació amb "surprise" i "happy", que són les dues amb millor rendiment, amb precisions de més del 75%.
Addicionalment, la matriu de confusió mostra que les emocions "fear" i "sad" són confoses sovint entre elles, així com també "neutral" i "sad" o "angry" i "sad", mentre que les emocions "happy" i "surprise" són rarament confoses amb alguna altra.
Finalment, amb aquest model que proporciona els millors resultats s'ha realitzat la predicció de 20 imatges mencionada prèviament, obtenint els següents resultats (15). La predicció és correcta en 13 de les 20 imatges, indicant un bon funcionament, encara que no ideal, del model.

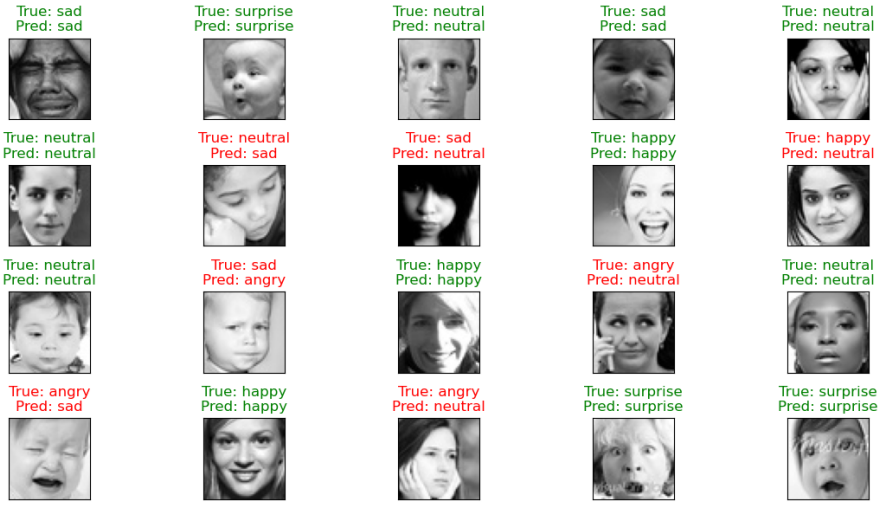


Figure 15. Predicció realitzada pel model final.

4.3. Intent de millora

Com s'ha mencionat en l'anterior secció, el model pel qual s'han obtingut els millors resultats mostra unes gràfiques (13) en què s'aprecia que hi pot haver cert *overfitting*. Per aquest motiu, aquesta última implementació es basa només en canviar el nombre de filtres de l'última capa convolucional de 1024 a 512, per tal d'intentar reduir aquest sobreajustament, ja que així es limita l'aprenentatge del model a característiques més senzilles.
No obstant, els resultats per a aquest intent de millora de la implementació (16), no són millors que els obtinguts anteriorment.

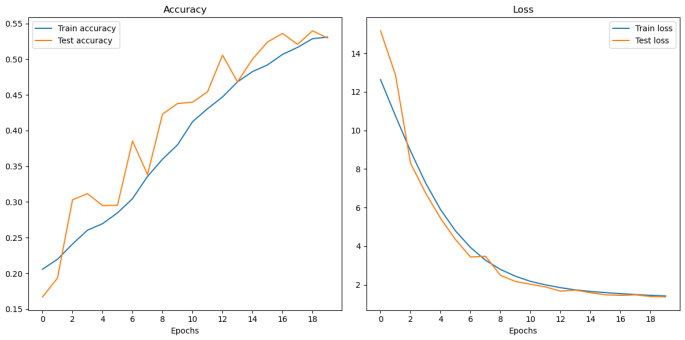


Figure 16. Gràfics per la precisió i pèrdua per al model corresponent a l'intent de millora.

La primera gràfica mostra que la diferència entre la precisió de *train* i *test* segueix sent molt similar, indicant que no s'ha reduit el sobreajustament com s'intentava. A més a més no s'obtenen millors valors ni per l'*accuracy* (53.09% en test) ni per la *loss*, de manera que aquest intent de millora no ha sigut efectiu i els millors resultats obtinguts durant el projecte són els mostrats en la secció anterior 4.2.

5. Conclusions, limitacions i possibles millores

En conclusió, el resultat del projecte ha acabat essent positiu, obtenint una precisió de 55.34%.
Aquesta precisió i els resultats generals que s'han mencionat anteriorment indiquen que el model pot predir de manera efectiva la categoria d'emoció de més de la meitat de les imatges que conté el *dataset*.
S'ha demostrat també la utilitat de les gràfiques de precisió i funció de pèrdua, la matriu de confusió i l'informe de classificació per a avaluar i analitzar el rendiment i l'eficàcia del model.

Les principals limitacions recauen en la diferència entre les diverses categories d'emocions presents en el dataset. Com s'ha comentat prèviament, algunes de les categories mostren resultats òptims amb precisions de quasi el 80% i d'altres molt pitjors resultats. A més diverses categories són molt més confoses entre elles que d'altres.
Per tant, la conclusió és que una millor qualitat de les imatges hagués suposat uns millors resultats per aquest projecte. D'aquesta manera, la limitació principal podria haver sigut evitada i les imatges presents en el *dataset* tinguessin la mateixa qualitat totes que les de les millors categories, o si més no, que fossin més específiques i diferents per a cada emoció.

En segon lloc, el temps ha representat també una limitació, ja que el temps d'execució és de 10 minuts per època, suposant unes tres hores totals.
Per a un projecte en què se li pogués dedicar molt més temps, es podria haver resolt el problema prèviament esmentat fent una tria de les millors imatges en cada categoria, millorant així el rendiment global.
Finalment, amb uns coneixements més elevats en el camp de la intel·ligència artificial podríem haver implementat un model amb una xarxa neuronal més sofisticada que hagués permès un rendiment superior.

References

[1] Munjal Sambare. *FER-2013 Dataset*. Accessed: 2024-12-15. URL: `https://www.kaggle.com/datasets/msambare/fer2013`.