Emotion, age, gender and ethnicity with CNN



Veton shala

Rahman Dogan

NBI/Handelsakademin

2024-12

# Abstract

This report explains how we use deep learning and computer vision to create a CNN model that is trained to classify emotions, age, gender and ethnicity in real-time using neural networks. The CNN models are trained on both training and validation data.

Innehållsförteckning

[Abstract 2](#_Toc184904954)

[1 Inledning 1](#_Toc184904955)

[2 Teori 2](#_Toc184904956)

[2.1.1 Förberedelser och bearbetning. 2](#_Toc184904957)

[2.1.2 Funktion för att ladda UTKFace dataset 2](#_Toc184904960)

[2.1.3 Laddar UTKFace dataset 3](#_Toc184904985)

[2.1.4 Laddar FER dataset 3](#_Toc184904986)

[2.1.5 Skapar träningsmodell 4](#_Toc184905012)

[2.1.6 Skapar FER modell 5](#_Toc184905047)

[2.1.7 Skapar UTK modell 5](#_Toc184905051)

[2.1.8 Kompilerar modellerna 5](#_Toc184905057)

[2.1.9 Callbacks 6](#_Toc184905081)

[2.1.10 Tränar FER modellen 6](#_Toc184905092)

[2.1.11 Tränar UTK modellen 6](#_Toc184905099)

[2.1.12 Laddar den bästa modellen 7](#_Toc184905122)

[3 Metod 8](#_Toc184905125)

[4 Resultat och Diskussion 9](#_Toc184905126)

[4.1 Modellens prestanda 9](#_Toc184905127)

[4.2 Realtidsapplikation 10](#_Toc184905127)

[5 Slutsatser 11](#_Toc184905128)

[Källförteckning 12](#_Toc184905129)

# Inledning

Convolutional Neural Networks är ett spännande område inom Artificiell intelligens och är inspirerad av hur hjärnan bearbetar visuell information. Syftet med denna rapport är att skapa en CNN-modell som kan klassificera ansiktsuttryck, ålder, kön och etnicitet.

För att uppfylla syftet kommer följande frågeställning att besvaras:

1. Hur kan CNN-modeller tränas för att klassificera ansiktsuttryck, kön, ålder och etnicitet.

# Teori

Convolutional Neural Networks(CNN)

### Förberedelser och bearbetning.

### folder\_path\_fer = "../input/face-expression-recognition-dataset/images/"

### folder\_path\_utk = "/kaggle/input/utkface-new/UTKFace"

picture\_size = 64   
batch\_size = 64

### Funktion för att ladda UTKFace dataset

### Def load\_utk\_data(folder\_path, target\_size=(64, 64)):

### images, ages, genders, ethnicities = [], [], [], []

### for img\_name in os.listdir(folder\_path):

### if img\_name.endswith('.jpg') or img\_name.endswith('.png'):

### img\_path = os.path.join(folder\_path, img\_name)

### img = load\_img(img\_path, target\_size=target\_size, color\_mode='rgb')

### img\_array = img\_to\_array(img) / 255.0

### parts = img\_name.split('\_')

### if len(parts) < 4:

### print(f"Skipping invalid file: {img\_name}")

### continue

### try:

### age = int(parts[0])

### gender = int(parts[1])

### ethnicity = int(parts[2])

### except ValueError:

### print(f"Skipping invalid file: {img\_name}")

### continue

### images.append(img\_array)

### ages.append(age)

### genders.append(gender)

### ethnicities.append(ethnicity)

### min\_length = min(len(images), len(ages), len(genders), len(ethnicities))

### return np.array(images[:min\_length]), np.array(ages[:min\_length]), np.array(genders[:min\_length]), np.array(ethnicities[:min\_length])

### Laddar UTKFace dataset

utk\_images, utk\_ages, utk\_genders, utk\_ethnicities = load\_utk\_data(folder\_path\_utk)

### Laddar FER dataset

### 

### datagen\_fer = ImageDataGenerator(

### rescale=1./255,

### rotation\_range=20,

### width\_shift\_range=0.2,

### height\_shift\_range=0.2,

### horizontal\_flip=True,

### validation\_split=0.2

### )

### fer\_train = datagen\_fer.flow\_from\_directory(

### folder\_path\_fer + "train",

### target\_size=(picture\_size, picture\_size),

### color\_mode="rgb",

### batch\_size=batch\_size,

### class\_mode='categorical',

### subset='training',

### shuffle=True

### )

### fer\_val = datagen\_fer.flow\_from\_directory(

### folder\_path\_fer + "train",

### target\_size=(picture\_size, picture\_size),

### color\_mode="rgb",

### batch\_size=batch\_size,

### class\_mode='categorical',

### subset='validation',

### shuffle=False

)

### Skapar träningsmodell

### def create\_model(input\_shape=(64, 64, 3)):

### inputs = Input(shape=input\_shape)

### x = Conv2D(64, (3, 3), padding='same', activation='relu')(inputs)

### x = BatchNormalization()(x)

### x = Conv2D(64, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = MaxPooling2D(pool\_size=(2, 2))(x)

### x = Dropout(0.3)(x)

### x = Conv2D(128, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = Conv2D(128, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = MaxPooling2D(pool\_size=(2, 2))(x)

### x = Dropout(0.3)(x)

### x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = Conv2D(256, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = MaxPooling2D(pool\_size=(2, 2))(x)

### x = Dropout(0.3)(x)

### x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = Conv2D(512, (3, 3), padding='same', activation='relu')(x)

### x = BatchNormalization()(x)

### x = MaxPooling2D(pool\_size=(2, 2))(x)

### x = Dropout(0.3)(x)

### x = Flatten()(x)

### x = Dense(512, activation='relu')(x)

### x = BatchNormalization()(x)

### x = Dropout(0.5)(x)

### x = Dense(256, activation='relu')(x)

### x = BatchNormalization()(x)

### x = Dropout(0.5)(x)

### return inputs, x

### Skapar FER modell

### fer\_inputs, fer\_features = create\_model()

### fer\_output = Dense(7, activation='softmax', name='expression\_output')(fer\_features)

### fer\_model = Model(inputs=fer\_inputs, outputs=fer\_output)

### Skapar UTK modell

### utk\_inputs, utk\_features = create\_model()

### age\_output = Dense(1, name='age\_output')(utk\_features)

### gender\_output = Dense(1, activation='sigmoid', name='gender\_output')(utk\_features)

### ethnicity\_output = Dense(5, activation='softmax', name='ethnicity\_output')(utk\_features)

### utk\_model = Model(inputs=utk\_inputs, outputs=[age\_output, gender\_output, ethnicity\_output])

### Kompilerar modellerna

### fer\_model.compile(

### optimizer=AdamW(learning\_rate=0.001, weight\_decay=1e-5),

### loss='categorical\_crossentropy',

### metrics=['accuracy']

### )

### utk\_model.compile(

### optimizer=AdamW(learning\_rate=0.001, weight\_decay=1e-5),

### loss={

### 'age\_output': 'mse',

### 'gender\_output': 'binary\_crossentropy',

### 'ethnicity\_output': 'categorical\_crossentropy'

### },

### loss\_weights={

### 'age\_output': 0.1,

### 'gender\_output': 0.1,

### 'ethnicity\_output': 0.1

### },

### metrics={

### 'age\_output': 'mae',

### 'gender\_output': 'accuracy',

### 'ethnicity\_output': 'accuracy'

### }

### )

### Callbacks

### fer\_callbacks = [

### EarlyStopping(monitor='val\_loss', patience=15, restore\_best\_weights=True),

### ModelCheckpoint('best\_fer\_model.keras', monitor='val\_accuracy', mode='max', save\_best\_only=True, verbose=1),

### ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=5, min\_lr=1e-6, verbose=1)

### ]

### utk\_callbacks = [

### EarlyStopping(monitor='val\_loss', patience=15, restore\_best\_weights=True),

### ModelCheckpoint('best\_utk\_model.keras', monitor='val\_loss', mode='min', save\_best\_only=True, verbose=1),

### ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=5, min\_lr=1e-6, verbose=1)

### ]

### 

### Tränar FER modellen

### fefer\_history = fer\_model.fit(

### fer\_train,

### epochs=100,

### validation\_data=fer\_val,

### callbacks=fer\_callbacks

### )

### Tränar UTK modellen

### utk\_train\_images, utk\_val\_images, utk\_train\_ages, utk\_val\_ages, utk\_train\_genders, utk\_val\_genders, utk\_train\_ethnicities, utk\_val\_ethnicities = train\_test\_split(

### utk\_images, utk\_ages, utk\_genders, utk\_ethnicities, test\_size=0.2, random\_state=42

### )

### utk\_history = utk\_model.fit(

### utk\_train\_images,

### {

### 'age\_output': utk\_train\_ages,

### 'gender\_output': utk\_train\_genders,

### 'ethnicity\_output': tf.keras.utils.to\_categorical(utk\_train\_ethnicities, num\_classes=5)

### },

### epochs=100,

### batch\_size=batch\_size,

### validation\_data=(

### utk\_val\_images,

### {

### 'age\_output': utk\_val\_ages,

### 'gender\_output': utk\_val\_genders,

### 'ethnicity\_output': tf.keras.utils.to\_categorical(utk\_val\_ethnicities, num\_classes=5)

### }

### ),

### callbacks=utk\_callbacks

### )

### Laddar den bästa modellen

### best\_fer\_model = load\_model('best\_fer\_model.keras')

### best\_utk\_model = load\_model('best\_utk\_model.keras')

# Metod

Arbetet har genomförts via Kaggle och med hjälp av primära ramverk som Keras/TensorFlow.   
Data (/kaggle/input/face-expression-recognition-dataset) och (<https://www.kaggle.com/datasets/jangedoo/utkface-new>) har erhållits direkt från Kaggle.

# Resultat och Diskussion

## Modellens prestanda

Nedan bilder visar att våra CNN-modeller kan klassificera ansiktsuttryck, ålder, kön och etnicitet.

CNN-modellerna för ansiktsanalys uppnådde bra resultat på både FER- och UTK-dataseten. FER-modellen som tränades för att känna igen ansiktsuttryck, uppnådde en valideringsnoggrannhet på cirka 67% efter 49. Epoker.

En bild som visar text, skärmbild, Teckensnitt, dokument

Automatiskt genererad beskrivning

UTK-modellen tränad för att förutsäga ålder, kön och etnicitet gav också bra resultat och utvärderas enligt följande:  
Åldersprediktion (age\_output): Modellen matar ut en regression och jämförs med det faktiska värdet.

Könsförutsägelse (gender\_output): Modellen returnerar ett värde mellan 0 (man) eller 1 (kvinna) med en sigmoidfunktion.

Etnicitetsprediktion (ethnicity\_output): Modellen förutsäger sannolikhetsfördelningen för 5 klasser (eventuellt specifika etnicitetskategorier) med hjälp av softmax-aktivering.

Noggrannhetstillstånd:

Noggrannhet för kön (gender\_output\_accuracy) och noggrannhet för etnicitet (ethnicity\_output\_accuracy) beräknas under träning.

För åldersuppskattning beräknas genomsnittligt absolut fel (mae) istället för noggrannhet.

En bild som visar text, Teckensnitt, nummer, skärmbild

Automatiskt genererad beskrivning

Dessutom hjälpte arkitekturen av modellerna som inkluderade flera faltningslager följt av batchnormalisering, maximal pooling och bortfall, att undvika överanpassning och förbättra generaliseringen.

Att använda återuppringningar som EarlyStopping och ReduceLROnPlateau bidrog till att optimera träningsprocessen och förhindra överanpassning.

## Realtidsapplikation

Realtidsexekvering, implementerad med React för frontend och Python för backend, möjliggör förutsägelser baserade på kamerabilder eller uppladdade foton. Denna funktion ger en praktisk och interaktiv användarupplevelse. Förutsägelsernas noggrannhet kan dock påverkas av faktorer som ljusförhållanden och bildkvalitet.

Framtida förbättringar kan fokusera på att förbättra modellens robusthet mot förändrade ljusförhållanden och bildkvaliteter. Detta kan uppnås genom att utöka träningsdatan med bilder tagna under olika förhållanden eller genom att tillämpa bildförbehandlingstekniker i realtidsapplikation.

# Slutsatser

Tränade CNN-modeller kan förutse ansiktsuttryck, ålder, kön och etnicitet.

En av de viktigaste utmaningarna i projektet var att använda två olika datamängder (FER och UTK) och att utbilda separata modeller för olika uppgifter. Framtida studier skulle kunna undersöka möjligheten att integrera dessa uppgifter i en enda multitasking-modell, vilket potentiellt kan förbättra den övergripande prestandan och effektiviteten.

Ytterligare förbättringar kan inkludera:

Finjustera modellarkitekturen för att förbättra noggrannheten, särskilt vid ansiktsigenkänning.  
Tillämpa mer avancerade dataökningstekniker för att öka modellens robusthet.  
Optimera modellen för mobila enheter för att möjliggöra bredare användning av applikationen.

Sammanfattningsvis demonstrerar projektet potentialen för djupinlärning i ansiktsanalys samtidigt som det lyfter fram utmaningarna med att bygga robusta modeller för realtidsapplikationer.

# Källförteckning

https://www.youtube.com/playlist?list=PLgzaMbMPEHEy33r5tgph8TFR\_\_UHuutBN https://app.datacamp.com/learn/courses/introduction-to-deep-learning-in-python https://app.datacamp.com/learn/courses/image-processing-with-keras-in-python https://1joyeducation.learnpoint.se/GroupForms/Group\_LearningContent\_Content.aspx?Id=3524 https://chatgpt.com