Rapport: Kunskapskontroll 2

Slutuppgift i kursen Deep Learning på folkhögskoleutbildningen Data Scientist på EC Utbildning AB i Göteborg.

Christoffer Olsson

# Introduktion

Uppgiften består i att undersöka en Deep Learning-modell för ansiktsuttryckigenkänning byggd av Akshit Madan (hädanefter kallad grundmodellen) och få den att fungera i egen miljö. Den här rapporten beskriver tillvägagångssättet.

## Bakgrund

Grundmodellen är en Deep Learning-modell tränad på bilder av ansikten med olika uttryck som analyserar en kameramatning i realtid och markerar upptäckta ansikten med en kvadrat och en etikett som anger ansiktsuttrycket.

## Syfte

Det är användbart att kunna utröna ansiktsuttryck på människor i många olika sammanhang. Det skulle till exempel kunna användas inom sjukvården i videosamtal med patienter för att se hur de reagerar på behandlingen. Ett annat användningsområde skulle kunna vara att övervaka förare av diverse fordon för att se huruvida de är trötta, alerta, fokuserade eller distraherade.

### Frågeställning

1. För att träna en modell på ansiktsigenkänning, vilken typ av neuralt nätverk behövs?
2. Vilken data finns tillgänglig att träna modellen på?
3. Vad är en rimlig precision i modellen?
   1. Kan jag bygga en bättre modell än grundmodellen?
4. Hur kan en överföra den tränade modellen till att läsa av en livematning från en kamera?
5. Vad finns det för sätt att förbättra modellen ytterligare?

# Databeskrivning / EDA (Exploratory Data Analysis)

Datasetet i projektet är "Face expression recognition dataset" av Jonathan Oheix och är taget från Kaggle[[1]](#footnote-28223). Det innehåller ett träningsset med 28 823 bilder och ett valideringsset med 7 066 bilder ojämnt fördelat på sju kategorier, som vardera representerar ett av följande ansiktsuttryck: “angry”, “disgust”, “fear”, “happy”, “neutral”, “sad” och “surprise”. Alla bilder är 48×48 pixlar stora och är i gråskala.

Det innehåller en stor mängd olika ansikten av alla åldrar, kön och utseenden hos människor och bilderna verkar vara tagna i olika naturliga situationer i varierande ljusa och mörka bakgrunder. Bilderna är beskurna så att ansiktet tar upp i stort sett hela filen. Jag kunde inte upptäcka några extremvärden eller defekta bilder i datasetet.

Grundmodellen använder funktionen ImageDataGenerator för att bygga dataseten av bilderna och eftersom de redan är indelade i ett tränings- och ett valideringsset behövs inte det göras i funktionen. Funktionen är dock förlegad och uppdateras till att istället tillämpa image\_dataset\_from\_directory.

Jag expanderade EDA-funktionen i grundmodellen till att visa sex bilder från varje kategori istället för nio från en, för att få en bättre överblick.

# Metod och Modeller (Teori)

För att träna en Deep Learning-modell på ansiktsigenkänning används lämpligen ett Convolutional Neural Network (CNN) som är väl utformat för bildigenkänning. Grundmodellen följer en relativt typisk arkitektur enligt koden nedan. Listorna med filter, kernel size och units visar värde för varje lager i stacken och varje paragraf är en stack (det vill säga fyra Conv2D-stackar och två hidden Dense-stackar):

model.add(Conv2D([64, 128, 512, 512], [3, 5, 3, 3], padding='same',

input\_shape=(48,48,1)))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=2))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense([256, 512]))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

Grundmodellen kompileras med categorical\_crossentropy som loss-funktion och Adam som optimerare. Slutligen fit:as modellen med tre callbacks: ModelCheckpoint, EarlyStopping och ReduceLROnPlateau.

Den sparade modellen laddas därefter in i ett Pythonskript som startar en kameramatning med ansiktsigenkänning som markerar ansikten med en kvadrat. Modellen analyserar kvadraten och ger en output om vilken kategori ansiktsuttrycket tillhör.

### Utveckling av modell

Det första steget var att testköra grundmodellen på en mindre del av datasetet. Det gick bra men den producerade flertalet varningar om förlegade Keras-moduler både relaterat till modellen och dataseten. Det introducerar svagheter som riskerar att påverka prestandan, vilket motiverar en omskrivning och utveckling av grundmodellen. Jag gör följande förändringar: Introduktion av validation\_split i dataseten för att reducera overfitting. Standardisering av datan att vara mellan 0 och 1 för att stabilisera träningsprocessen; introduktion av funktioner för cache och prefetch för att förbättra träningsprestandan[[2]](#footnote-15324); tillämpning av KerasTuner och RandomSearch för att förbättra arkitekturen.

Jag skriver dessutom en helt egen modell i syfte att testa om det går att uppnå högre precision med en annan arkitektur. Totalt görs tre iterationer av modellen med ökande komplexitet och i den slutgiltiga genomsöks hyperparametrarna mängden Conv2D-lager, Dense-lager och antalet filter/units, om den skulle använda MaxPooling2D och Dropout, samt bästa learning\_rate. De två sista iterationerna inkluderar ett Conv2d-lager med kernel\_size=7 och stride=2 som input-lager.

Den bästa arkitekturen (iteration 2):

model.add(Conv2D([32, 112, 112, 112], [7, 3, 3, 3], strides=[2, 1, 1, 1],

padding='same', activation='relu', input\_shape=(48,48,1)))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=2))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense([384, 384, 448], activation='relu'))

model.add(BatchNormalization())

model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

Ett försök görs också att träna om med grundmodellens arkitektur med tune:ad learning\_rate och de uppdaterade dataseten. Det är dock ett stort problem att grundmodellen är tränad på samma dataset, vilket kan leda till overfitting och det här steget görs främst i demonstrationssyfte att ladda in en modell för att bygga vidare på.

# Projekt Resultat och Analys

Följande tabell visar resultaten av en evaluering på valideringssetet på samtliga testade modeller. Den tune:ade är med i demonstrationssyfte med brasklappen att den är tränad två gånger på samma dataset.

|  |  |  |
| --- | --- | --- |
| **Modell** | **Loss** | **Accuracy** |
| Grundmodellen | 0.9643 | 0.6575 |
| Grundmodellen, tune:ad | 0.9120 | 0.7141 |
| Egen modell, iteration 1 | 1.1341 | 0.6023 |
| Egen modell, iteration 2 | 1.2150 | 0.6143 |
| Egen modell, iteration 3 | 2.3674 | 0.6008 |

Grundmodellens accuracy borde vara ett bra riktmärke. Vid test av hela programmet märks att den för det mesta upptäcker ett korrekt ansiktsuttryck, särskilt när en bild med ett ansikte visas framför kameran. Den är ibland förvirrad över ett rörligt ansikte och blandar ihop olika uttryck men det går som regel att “forcera” fram korrekt klassificering genom att ändra sin min något. Samma sak gäller dock iteration 2 av min modell som i praktiken har snarlik prestanda (bättre med vissa ansiktsuttryck som “disgust” men sämre med andra som “surprise”) trots att den har lägre accuracy.

Den mest komplexa iteration 3 av den egna modellen visar sig vara för komplex och lider av overfitting och sämre accuracy på valideringssetet.

# Slutsats och förslag på potentiell vidareutveckling

Metoden beskriven i rapporten producerar ett rimligt träffsäkert program för ansiktsuttryckigenkänning. Den har svårare för att känna igen vissa uttryck framför andra och tenderar att förväxla snarlika uttryck. Att de olika modellerna tenderar att ha svårt för olika ansiktsuttryck tyder på att träningsprocessen inte är optimerad. För att göra modellen användbar i produktion behövs mer arbete med den.

I förhållande till träningstid är iteration 2 överlägsen alla andra och tog cirka 1/10 av tiden jämfört med grundmodellen och iteration 3 av min modell. Om tid och beräkningskraft är en bristvara är den ett gott alternativ (förutsatt att man inte har tillgång till en förtränad modell).

Dataseten skulle kunna ha mer högupplösta bilder och vara i färg. Det skulle förmodligen vara gynnsamt med data augumentation, särskilt för klasserna som har färre bilder.

Modellen skulle kunna bli bättre om fler hyperparametrar inkluderades i GridSearch. Alla föreslagna lösningar kräver dock betydande mer beräkningskraft men eftersom ansiktsigenkänning är ett stort område vore det rimligt att det finns ännu bättre färdiga modeller som kunde laddas in i pre-training-syfte som skulle kunna förenkla processen.

Redogörelse

## Utmaningar du haft under arbetet samt hur du hanterat dem.

De största problemen jag stötte på var de förlegade paketen i grundmodellen. I slutändan innebar de att jag skrev om i stort sett hela koden. Det hade även smugit sig in ett par missar, t.ex. Callback-klassen ReduceLROnPlateau i grundmodellen har samma patience som EarlyStopping, vilket innebär att de aktiveras samtidigt och modellen hinner inte träna med reducerad LR och gör därmed ingenting. Ökad patience på EarlyStopping löste problemet. Det var dessutom flertalet importerade bibliotek som inte användes och jag behövde gå igenom alla importer grundläggande för att städa undan.

Med första iterationen hade min modell högre accuracy än grundmodellen men vid praktisk användning presterade den märkbart sämre. Problemet försvann i nästa iteration.

Till sista iterationen av modellen gick väldigt mycket tid åt till träningen. Jag skulle ha implementerat EarlyStopping under RandomSearch då loggen visade att de flesta trials var dömda att misslyckas tidigare än antalet epoker de körde igenom.

## Vilket betyg du anser att du skall ha och varför.

Att identifiera problemen med grundmodellen och applicera en fullständig omskrivning av funktionerna för att skapa dataseten, introducera KerasTuner och skriva flera funktioner för att justera ett antal hyperparametrar och lyckas träna en originell modell visar att jag självständigt kan vidareutveckla arbetsprocesser inom Deep Learning på ett utförligt sätt och därför anser jag att jag borde tilldelas VG.

## Tips du hade ”gett till dig själv” i början av kursen nu när du slutfört den.

* Bekymra dig inte så mycket över att förstå all teoretiskt direkt från början, det är bättre att börja jobba tidigare direkt med kod för det hjälper förståelsen.
* Ha kul och våga experimentera. Det finns jättebra verktyg för att kontrollera hur väl modeller presterar.
* Det finns inget "perfekt" sätt att lösa en uppgift på, använd fantasin och iterera modellerna tills du är nöjd.

1. <https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset> [↑](#footnote-ref-28223)
2. https://www.tensorflow.org/tutorials/load\_data/images [↑](#footnote-ref-15324)