Hockey vs Tennis CNN

Frikk Myhre Slåsletten 02.11.2025

1. Problemet

Omfang/Scope

I dette prosjektet var formålet å utvikle et system som automatisk kunne klassifisere bilder som enten «hockey» eller «tennis». Prosjektet kombinerer maskinlæring med et webgrensesnitt i Java Spring, hvor brukeren kan laste opp bilder av enten ishockey eller tennis, direkte og få en visuell og tekstlig prediksjon på hva som er lastet opp.

Modellen brukt er et konvolusjonelt nevralt nettverk som selv lærer å identifisere mønstre som skiller bildene.

Jeg har i dette prosjektet tatt inspirasjon fra et tidligere, selvlaget prosjekt som klassifiserte bilder relatert til jobbsøking. Dette tidligere prosjektet er også publisert i GitHub, og fokuserte kun på bildeklassifiseringen uten bruker-input. I denne versjonen er det tidligere systemet videreutviklet til å ta brukerinput, slik at modellen kan ta imot opplastede bilder direkte og gi resultat i sanntid.

Brukere og nytteverdi

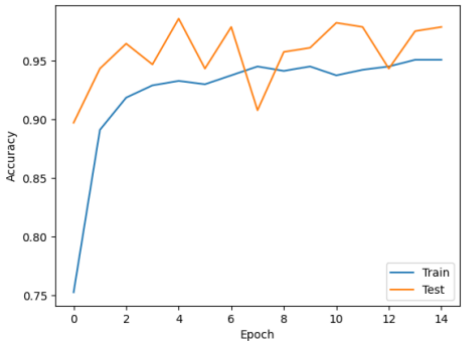
Her i mitt prosjekt er brukeren en som ønsker å teste eller demonstrere CNN-modeller gjennom et lett webgrensesnitt. Systemet visualiserer enkelt og gjør det lett å tolke ML-modeller uten å kjøre kode direkte.

Ressurser

Systemet kjører lokalt, med trening og testing gjort i Python med TensorFLow. Modellfilene *model.h5* og *labels.json* brukes av FastAPI-server som kommuniserer med Java Spring.

Metrikker

Ytelsen er målt i accuracy på treningsdata og testdata. Modellen oppnår regelrett test accuracy på omtrent 95-96% etter 15 epoker. I terminalen under kjøring får vi også opp *loss* for modellen i back-end, men dette kommer ikke frem front-end. Ved opplasting av bilde i front-end får bruker tilbake hvor sikker modellen er, igjen i accuracy.



2. Data

Datasettet består av bilder fordelt på to klasser, hockey og tennis. Bildene ble hentet fra Kaggle og lagt i mapper for train/ og test/. Dataen er fargede bilder skalert til 64x64 piksler for at jeg skulle kunne være sikker på at de har samme bredde og høyde. I tillegg ble dette gjort for å redusere minne og prosessorkraft. Jeg tok også en vurdering på at det var nok detaljer i 64x64 for å få frem det jeg trengte.

3. Modellering

Modellen er et Convolutional Neural Network (CNN) implementert i TensorFlor/Keras. Det består av følgende kovolusjonslag, forenklet:

Conv2D → MaxPooling → Conv2D → MaxPooling → Flatten → Dense → Dropout → Output

Conv2D oppdager mønstre i bildet, *MaxPooling* reduserer dimensjonene som tidligere forklart, igjen *Conv2D* lærer nye mønstre på lavere nivå av detaljer, *MaxPooling2D* igjen for å redusere dimensjonen ytterligere for å fokusere på de aller viktigste trekkene i bildet. *Flatten* for å gjøre om til vektor. *Dense* som gjør at vi lærer sammenhengen mellom features og klassene. Avslutter med *Dropout* for å tvinge nettverket til å lære mønstre istedenfor å huske treningsdata.

4. Deployment

Etter at modellen ble trent i Python, ble den tilgjengeliggjort med FastAPI. FastAPI gir meg endepunktet /predict som tar imot bilder og returnerer resultatet i JSON.

Spring Boot fungerer som frontend hvor brukeren kan laste opp bildet via webgrensesnittet, hvor forespørselen går til FastAPI-serveren. Videre forbedring kan være å deploye FastAPI til en ekstern server og for eksempel legge til så mange idretter at man nesten bare kan laste opp hva som helst.

Denne løsningen demonstrerer hele livssyklusen til ML-prosjektet, fra datainnsamling og modellering til publisering og interaksjon.

5. Referanser

TensorFlow/Keras: https://tensorflow.org
FasatAPI: https://fastapi.tiangolo.com
Datasett: https://www.kaggle.com/datasets/rishikeshkonapure/sports-image-dataset