Teoretiska frågor

1. Deep learning är en del av machine learning som är en del av AI.

2. Tensorflow beskrivs ofta som motorn och Keras som ratten. Där Tensorflow hanterar beräkningarna för att träna neurala nätverk och Keras är lite mer användarvänligt med olika "byggstenar" som kan kombineras ihop till olika typer av neurala nätverk. Dessa "byggstenar" kan t.ex. vara olika lager med olika aktiveringsfunktioner, optimerare och loss functions.

3. Parametrar: är variabler som modellen lär sig från under träning och exempel på det är t.ex. olika vikter och bias.
   Hyperparametrar: kan ses som inställningar för sin modell som man gör innan man tränar den. Dessa styr hur modellen kommer lära sig. Hur dessa väljs kommer kunna ha en stor påverkan på hur bra och effektiv träningen av modellen blir.

4. Träningsdata: Används för att träna sin modells olika parametrar .
   Valideringsdata: Snvänds under träningen som data som modellen inte har sett att testa mot. Detta används för att undvika överanpassning och för att man ska kunna göra justeringar på sina hyperparametrar för att på så sätt hitta den bästa modellen för sitt dataset.
   Testdata: Används för att testa sin modells generaliseringsförmåga på helt ny osedd data. Denna data måste alltså vara helt osedd för modellen sedan tidigare för att kunna få en så bra bedömning som möjligt av modellen generaliseringsförmåga på ny data.

5. Detta är ett neuralt nätverk som tränas.
   - Till att börja med läses in hur många kolumner som finns i träningsdatan.
   - Sedan läggs ett denslager till med 100 neuroner som har aktiveringsfunktionen relu och vi talar om hur många numeriska värden som lagret kommer att ta emot.
   - Därefter läggs ett dropout-lager till som slumpmässigt sätter i genomsnitt 20 % av signalen in till lagret till 0. Detta används för att göra modellen mer robust och för hindra överanpassning.
   - Ett nytt denslager läggs till med 50 neruroner och relu som akriveringsfunktion.

- Ett slutligt denslager med 1 neuron och sigmoid aktivering som är vanligt för binär klassificering där utdatan är mellan 0 och 1).
- nn_model.compile konfigurerar hur det neurala nätverkat ska tränas. Optimeringsfunktion är Adam, loss-funktionen som modellen försöker minimera under träning är binary cross-entropy som är ett vanligt val vid binär klassificering och accuracy används för att utvärdera modellens prestanda under träning och test.
- Sedan kommer modellen att tränas med 20 % av träningsdatan som valideringsdata och en early stopping som stoppar träningen om valideringsförlusten inte har förbättrats under 5 epoker på rad.

6. Regularisering är till för att se till så att modellen man tränar inte ska bli överanpassad. Olika regulariseringstekniker lägger till en begränsning eller straff som gör att modellen hålls enklare och inte blir överpassad och därmed bättre på ny osedd data.

7. Dropout är en regulariseringsteknik som tillfälligt slår av en slumpmässigt utvald nod under träningen. Detta gör att modellen mer robust eftersom den inte kan förlita sig på att alla noder alltid finns vilket gör att andra noder måste hjälpa till och "ta ansvar" i inlärningsprocessen.

8. Early stopping är en regulariseringsteknik som går ut på att träningen avbryts i förtid om prestandan på valideringsdatan blir sämre under träning under ett visst antal epoker (som bestäms av patience).

9. CNN - Convolutional Neural Networks

10. Översiktiligt kan man säga att ett CNN fungerar så att det kan lära sig att "se" mindre detaljer (t.ex. horisontella linjer eller hörn och kanter) som sedan kan sättas ihop till allt större former. Först kan det "se" kanter, sedan ser det öron, näsa, mun, där efter kan det sätta ihop det till ett ansikte och slutligen bestämma att det är t.ex. en katt. Allt detta sker med hjälp av convolutional-lager som har små filter (t.ex. 3x3 pixlar) i sig som letar efter t.ex. kanter. Sedan används pooling lager som förminskar bilden genom att t.ex. plocka ut maxvärdet ur ett visst antal pixlar. På detta sätt behålls de viktigaste delarna i bilden och det snabbar på beräkningen för kommande lager. I ett CNN brukar man ha flera convolutional och pooling lager efter varandra och efter dem kommer det ett antal dens-lager som används för klassificering och bestämmer om det t.ex. är en katt eller något annat.

11. Model.save sparar den färdigtränade modellen med alla dess viktade parametrar till en fil med namnet model_file.keras.

    Load_model laddar in den sparade neurala nätvärksmodellen från model_file.keras och lagra den i variabelen my_model. Sedan kan man använda my_model för att göra prediktioner utan att man behöver träna om modellen på nytt.

12. CPU: Central Processing Unit, är en generell processor som är bra på att hantera en bred variation av uppgifter snabbt, men är begränsad när det gäller att hantera många parallella beräkningar samtidigt.

    GPU: Graphics Processing Unit, är specialiserad på att utföra många enkla beräkningar parallellt, vilket gör den extremt effektiv för uppgifter som kräver hög parallellisering, som matrisoperationer i neurala nätverksträning.

Diskussion kring hur modellen kan användas i verkligheten.

RAG-chatbot, specialiserad på brädspelet Terraforming Mars

Som modellen och appen är nu (den finns på terraformingmars.streamlit.app) så kan den användas som ett hjälpmedel för att tolka/förklara regler och hur olika projekt kort fungerar. Man kan på ett lätt och snabbt sätt få tillgång till denna information istället för att behöva slå i regelböcker. På detta sätt kan man snabba på spelets gång om det skulle uppstå frågor och funderingar.

Utvecklarna av spelet skull kunna implementera denna chatbot på sin hemsida som en extra service. Det är väl här det skulle kunna gå att tjäna några pengar på chatbotten genom att sälja den till skaparna av spelet dock inte troligt för jag vet att de håller i pengarna 😊

Utmaningar och förbättringar är att alla expansionskort inte finns med i databasen. Det går så klart att lägga till men kommer att kräva relativt mycket manuellt arbete. Här skulle det vara lättare om man fick tillgång till utvecklarnas data direkt och kanske få det på ett bättre format redan från början. Ytterligare utmaningar blir att så fort något uppdateras (kort, regler eller nya expansioner) så behöver databasen uppdateras för att chatbotten ska ha tillgång till allt material. Chatbotten lyckas inte heller svar på alla frågor, kanske har det att göra med hur den är promtad eller så kanske det behövs en mer resonerande modell. Just nu är den också tillsagd att vara mycket kort och koncis för att bara ge de direkta svaren utan annat fluff. Så en utmaning är att den ska svara tillräckligt bra för att man ska vilja använda den annars förlorar den hela sitt syfta.

Ytterligare en fundering är hur man ska göra med API:nycklar. Om det skulle bli en mycket stor användning av chatboten får man fundera på om användarna måste skaffa sig en egen API-nyckel för att få tillgång till chatboten.

Några slutliga förbättringsåtgärder som man kan göra är att chatboten ska ha minne från tidigare ställda frågor, visa vart i materialet som den har hittat sin information och att den skulle kunna tolka och förklara bilder från regelböcker och kort.

Självutvärdering

1. Roligast tycker jag är att man skapar något som faktiskt är användbart och att man kan se potential för att det kan användas inom många olika områden.

2. Jag hoppas på VG, tycker jag har fått med de moment som efterfrågats.

3. Mest utmanade har tråkigt nog varit att försöka få till en bakgrundsbild i streamlit, som jag fortfarande inte har lyckats få så som jag vill. Så det struntar jag i nu! Annars har det varit utmanade att få till chunkingen så den funkar som den ska. Det finns även många fler sätt att chunka än vad jag har testat, även om jag tyckte det funkade bra på det sätt jag gjorde så finns det säkert förbättringspotential där också.