Teoretiska frågor

1. Hur är AI, Maskininlärning och Deep Learning relaterat?

Artificiell intelligens (AI) är ett samlingsbegrepp för tekniker som gör det möjligt för datorer att efterlikna mänskligt tänkande och fatta beslut utan att vara direkt programmerade för varje enskild uppgift. Inom detta område spelar maskininlärning en central roll. Maskininlärning innebär att datorn tränas på historisk data för att kunna identifiera mönster och göra förutsägelser på ny, osedd data.

En särskilt avancerad form av maskininlärning är Deep Learning. Den bygger på artificiella neurala nätverk med flera lager, kan ses att försöka efterlikna en hjärna, där varje lager successivt tar an informationen som modellen får. Detta gör Deep Learning särskilt användbart vid hantering av komplex och ostrukturerad data, såsom bilder, ljud och text. Genom att automatiskt lära sig relevanta mönster direkt från rådata kan dessa modeller nå mycket hög accuracy, särskilt när stora mängder data finns tillgängligt. Deep Learning kan därmed ses som en underkategori inom maskininlärning, som i sin tur är en del av det bredare AI-fältet.

2. Hur är TensorFlow och Keras relaterat?

TensorFlow är ett omfattande ramverk för maskininlärning och Deep Learning, utvecklat av Google. Keras är ett högre nivå-gränssnitt för att bygga och träna neurala nätverk och fungerar som en förenklad API ovanpå TensorFlow. Keras gör det enklare att snabbt bygga modeller genom att ge en användarvänlig strukur, medan TensorFlow ger tillgång till mer avancerad kontroll och prestandaoptimering.

3. Vad är en parameter? Vad är en hyperparameter?

En *parameter* är en intern variabel i en modell som lärs under träningen, till exempel vikter och biasar i ett neuralt nätverk. En *hyperparameter* är däremot en inställning som styr träningsprocessen och måste väljas innan träningen börjar. Exempel på hyperparametrar är inlärningshastighet, antal lager, antal neuroner per lager och dropout-rate.

4. Förklara tränings-, validerings- och testdataset.

- Träningsdata används för att lära modellen att identifiera mönster.
- Valideringsdata används för att justera hyperparametrar och för att avgöra när träningen bör stoppas (t.ex. vid early stopping), utan att överanpassa till träningsdatan.
- Testdata används i slutset för att objektivt utvärdera modellens förmåga på ny, osedd data.

5. Förklara vad koden gör:

```
n_cols = x_train.shape[1]

nn_model = Sequential()
nn_model.add(Dense(100, activation='relu', input_shape=(n_cols, )))
nn_model.add(Dropout(rate=0.2))
nn_model.add(Dense(50, activation='relu'))
nn_model.add(Dense(1, activation='sigmoid'))

nn_model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy' ])

early_stopping_monitor = EarlyStopping(patience=5)
nn_model.fit(
    x_train,
    y_train,
    validation_split=0.2,
    epochs=100,
    callbacks=[early_stopping_monitor])
```

Koden skapar och tränar ett neuralt nätverk för binär klassificering med hjälp av Keras. Först identifieras antalet indatafunktioner från träningsdatan med hjälp av x_train.shape[1], vilket används för att definiera nätverkets indataformat. Därefter konstrueras ett sekventiellt nätverk bestående av flera lager. Det första lagret är ett Dense-lager med 100 noder och ReLU-aktiveringsfunktion, följt av ett Dropout-lager med en dropout-rate på 0,2, vilket innebär att 20 procent av neuronerna slumpmässigt inaktiveras vid varje träningssteg för att minska risken för överanpassning. Efter detta följer ytterligare ett Dense-lager med 50 noder och ReLU-aktivering. Utgångslagret består av en enda nod med sigmoid-aktivering.

Modellen kompileras med "adam" som optimeringsalgoritm och "binary_crossentropy" som lossfunktion, vilket är ett vanligt val vid binära klassificeringsproblem. Lossfunktionen används för att mäta hur långt modellens prediktioner ligger från de faktiska målen i träningsdatan. Genom att minimera värdet på lossfunktionen lär sig modellen att göra mer korrekta förutsägelser över tid. För att undvika överträning används early stopping, en regulariseringsteknik som övervakar validation loss under träningen. Om ingen förbättring sker under fem epoker avbryts träningen i förtid. Slutligen tränas modellen på träningsdatan med 20 procent avsatt för validering, och antalet epoker är max 100, men träningen kan avslutas tidigare om early stopping aktiveras.

6. Vad är syftet med att regularisera en modell?

Syftet med regularisering är att minska risken för överanpassning (overfitting), det vill säga att modellen lär sig för mycket av träningsdatan – inklusive brus – och därmed presterar sämre på ny data. Regularisering hjälper modellen att generalisera bättre.

7. Vad är Dropout?

Dropout är en regulariseringsteknik där vissa neuroner slumpmässigt "stängs av" under varje träningssteg. Det innebär att nätverket inte blir beroende av enskilda neuroner utan lär sig mer robusta och generaliserbara mönster. Under testning används hela nätverket, men vikterna justeras baserat på dropout-frekvensen.

8. Vad är Early Stopping?

Early Stopping är en teknik där träningen avslutas i förtid om modellen slutar förbättras på valideringsdatan. Detta förhindrar att modellen fortsätter träna och överanpassar till träningsdatan. Det är ett sätt att automatiskt avgöra ett lämpligt antal epoker.

9. Populärt neuralt nätverk för bildanalys?

För bildanalys är Convolutional Neural Networks (CNNs) den mest populära nätverkstypen. CNNs är särskilt effektiva för att hantera bilddata eftersom de kan identifiera rumsliga mönster som kanter, former och objekt på olika nivåer av komplexitet.

10. Hur fungerar ett Convolutional Neural Network (CNN)?

Ett CNN fungerar genom att använda konvolutionslager som applicerar filter på indata för att extrahera viktiga egenskaper, som kanter eller färgvariationer. Dessa följs ofta av pooling-lager, som reducerar datans dimensioner och gör modellen mer effektiv. Efter flera lager extraheras allt mer abstrakta egenskaper, och slutligen används fullt anslutna (dense) lager för klassificering. Genom att använda viktdelning och lokal anslutning kan CNNs hantera stora bildinmatningar med relativt få parametrar.

11. Vad gör koden?

```
model.save("model_file.keras")
my_model = load_model("model_file.keras")
```

Koden sparar en tränad modell till en fil med namnet model_file.keras. Därefter laddas modellen från samma fil tillbaka till variabeln my_model, vilket gör det möjligt att använda modellen senare utan att behöva träna om den.

12. Vad är CPU och GPU?

CPU (Central Processing Unit) är datorns huvudsakliga processor och är optimerad för att hantera ett brett spektrum av uppgifter i följd.

GPU (Graphics Processing Unit) är optimerad för att utföra många operationer parallellt, vilket gör den särskilt effektiv för att träna deep learning-modeller, där beräkningarna är mycket intensiva och kan parallelliseras. Plattformar som Google Colab erbjuder tillgång till GPU för att påskynda träningen av modeller.