Camilla Dahlman

Kunskapskontroll 2
DS24 Deep Learing

Teoretiska frågor

1. Hur är AI, Maskininlärning och Deep Learning relaterat?
AI, artificiell intelligens, är det övergripande begreppet – det handlar om tekniker som får datorer att utföra uppgifter som normalt kräver mänsklig intelligens, som att förstå språk, känna igen mönster eller fatta beslut.

Maskininlärning (machine learning) är en delmängd av AI. Här handlar det om att skapa modeller som kan lära sig av data – utan att man talar om exakt hur den ska lösa varje uppgift. Modellen förbättras ju mer data den får jobba med.

Deep learning är en ännu mer specifik del av maskininlärning. Den bygger på så kallade artificiella neurala nätverk – inspirerade av hur den mänskliga hjärnan fungerar. Här används avancerade nätverksarkitekturer som CNN (för bildanalys), RNN (för sekvensdata som text eller ljud) och Transformer (som används i t.ex. språkmodeller som ChatGPT). Deep learning kräver mycket beräkningskraft vilket skiljer det från mer traditionella ML-metoder.

2. Hur är Tensorflow och Keras relaterat?
TensorFlow är ett ramverk för maskininlärning som erbjuder stor flexibilitet och beräkningskraft.
Keras är ett användarvänligt verktyg, fullt integrerat med TensorFlow som gör det lättare att skapa och träna modeller. Det innehåller färdiga komponenter som gör att man slipper skriva all kod från grunden.

3. Vad är en parameter? Vad är en hyperparameter?
Parametrar är de värden som modellen lär sig från data under träningen. Dessa uppdateras iterativt genom optimeringsalgoritmer.

Hyperparametrar styr hur modellen tränas och struktureras, till skillnad från parametrar som lärs in automatiskt från data. Hyperparametrar måste ställas in manuellt eller optimeras genom systematiska experiment. Valet av hyperparametrar har ofta avgörande betydelse för modellens prestanda, generalisering och träningshastighet. Exempel på hyperparametrar i neurala nätverk är antal dolda lager.

4. När man skall göra modellval och modellutvärdering kan man använda tränings-, valideringss- och testdataset. Förklara hur de olika delarna kan användas.

Träningsdatan används för att bygga och träna olika modeller. Det är alltid möjligt att skapa en modell som passar träningsdatan perfekt, men detta är inte målet. Istället strävar vi efter en modell som kan hantera ny, osedd data genom att generalisera inlärningen.

Valideringsdatan används för att utvärdera olika modeller och välja den bästa. Dessutom hjälper den till att förhindra overfitting, när en modell lär sig specifika detaljer i träningsdatan istället för att identifiera övergripande mönster. Om vi väljer en modell enbart baserat på träningsdatan finns risken att den fungerar dåligt på nya data. Valideringsdatan används ofta vid tuning av hyperparametrar.

Testdatan används för att utvärdera den modell som valts efter valideringssteget. Detta test görs endast en gång.

5. Förklara vad nedanstående kod gör:

```
# Definierar antalet features (kolumner) i x_train och sparar i variabeln n_cols
n_cols = x_train.shape[1]

# Skapar en sequential modell
nn_model = Sequential()

# Definerar det första lagret:
# Lägger till ett Dense-lager med 100 neuroner och ReLU-aktivering
# input_shape anger att detta lager förväntar sig lika många ingångar som antalet features i x_train
nn_model.add(Dense(100, activation='relu', input_shape=(n_cols, )))

# Lägger till funktionen drop out där det är 20% sannolikhet per neuron att blir droppad.
nn_model.add(Dropout(rate=0.2))

# Lägger till ett andra Dense-lager med 50 neuroner och ReLU-aktivering
nn_model.add(Dense(50, activation='relu'))

# Lägger till ett utgångslager med 1 neuron och sigmoid-aktivering för binär klassificering
nn_model.add(Dense(1, activation='sigmoid'))

# Förbereder modellen för träning och väljer optimerare (’adam’)
nn_model.compile(
        optimizer='adam',
        # Lägger till förlustfunktionen crossentropy för (binär klassificering)
```

```
        loss='binary_crossentropy',
        # Lägger till accuracy som ett mått för att mäta hur ofta modellen
        klassificerar korrekt
        metrics=['accuracy' ])

# Stoppar efter 5 epoker om resultatet inte förbättrats
early_stopping_monitor = EarlyStopping(patience=5)
nn_model.fit(
        x_train,
        y_train,
        validation_split=0.2,
        epochs=100,
        callbacks=[early_stopping_monitor])
```

6. Vad är syftet med att regularisera en modell?
Regularisering handlar om att **begränsa modellens komplexitet** för att **förbättra generalisering** (hur bra den funkar på ny data). Det motverkar också överanpassning.

7. ”Dropout” är en regulariseringsteknik, vad är det för något?
Drop out metoden slumpar bort noder under varje träningssteg. Det bidrar till att modellen inte övertränas. “Nätverket kan inte bli för beroende av enskilda noder”.

8. ”Early stopping” är en regulariseringsteknik, vad är det för något?
En funktion som gör att träningen stoppas när modellen inte längre förbättras (när den körs på valideringsdatan). Det förhindrar överträning och gör att träningen snabbas upp.

9. Din kollega frågar dig vilken typ av neuralt nätverk som är populärt för bildanalys, vad svarar du?
För bildanalys används CNN modeller. De är särskilt populära för uppgifter som bildklassificering och objektdetektering.

10. Förklara översiktligt hur ett ”Convolutional Neural Network” fungerar.
Ett Convolutional Neural Network (CNN) känner igen mönster i bilder stegvis.
Först hittar det enkla saker som linjer och färgövergångar (low-level features), och sedan mer komplexa delar som tex ögon och munnar (high-level features).
Modellen använder convolutional layers med filter som letar efter olika bildmönster.
Dessa följs ofta av pooling layers som minskar bildens storlek och fokuserar på det viktigaste.
För att täcka hela bilden jämnt används ibland padding.
Till slut används de upptäckta mönstren för att göra en prediktion, t.ex. om det är en tiger eller inte.

11. Vad gör nedanstående kod?

```
# Sparar modellen, både arkitektur och vikter, till en fil med namnet model_file.keras
model.save("model_file.keras")
# Läser in modellen från filen och skapar en instans som sparas i my_model
my_model = load_model("model_file.keras")
```

Användbart för att kunna återanvända modellen efter det att den tränats.

12. Deep Learning modeller kan ta lång tid att träna, då kan GPU via t.ex. Google Colab skynda på träningen avsevärt. Skriv mycket kortfattat vad CPU och GPU är.

CPU (Central Processing Unit) är datorns allmänna processor som hanterar olika typer av instruktioner.

GPU (Graphics Processing Unit) är effektiva vid parallella beräkningar och används ofta istället för CPU:er vid träning av deep learning-modeller. GPU:er är bra på matrisberäkningar, vilket är ytterligare en anledning till att de passar för Deep Learning.