

# Generativ AI og Store Språkmodeller

## Momentliste G01–G14

ELMED219 / BMED365

Universitetet i Bergen

Våren 2026

# Oversikt

- 1 Introduksjon til Generativ AI
- 2 Transformer-arkitekturen
- 3 LLM-grunnleggende
- 4 Prompt Engineering
- 5 Utfordringer med LLM
- 6 Modeller og avanserte konsepter

# G01: Definere generativ AI og skille fra diskriminativ AI

## Diskriminativ AI:

- Lærer å **klassifisere** eller skille mellom kategorier
- Modellerer  $P(y|x)$  – sannsynlighet for klasse gitt input
- Eksempler: "Er dette kreft?", "Hvilken sykdom?"
- CNN for bildeklassifisering

## Generativ AI:

- Lærer å **skape** nytt innhold
- Modellerer  $P(x)$  eller  $P(x|y)$  – datafordelingen selv
- Eksempler: Tekst, bilder, kode, musikk
- LLM, diffusjonsmodeller

## Nøkkelforskjell

**Diskriminativ:** Input → Kategori/Label

**Generativ:** Prompt/Instruksjon → Nytt innhold

## G02: Forklare self-attention-mekanismen på et konseptuelt nivå

**Self-attention:** “Hva er relevant for hva?”

**Intuisjon:**

- For hvert ord: Se på **alle** andre ord i setningen
- Beregn en **vektet sum** basert på relevans
- Fanger opp avhengigheter over lange avstander

**Eksempel:**

“Pasienten tok **medisinen**, og **den** hjalp mot smertene.”

- Self-attention kobler “den” til “medisinen” (ikke “smertene”)
- Modellen lærer hvilke koblinger som gir mening

### Query, Key, Value (QKV)

Hvert token genererer tre vektorer:

**Query:** “Hva leter jeg etter?” | **Key:** “Hva kan jeg tilby?” | **Value:** “Hva er innholdet mitt?”

# G03: Beskrive transformer-arkitekturen (encoder-decoder)

## Transformer-arkitekturen (Vaswani et al., 2017):

### Encoder:

- Prosesserer input-sekvens
- Bygger kontekstrik representasjon
- Brukes i: BERT, embeddings

### Decoder:

- Genererer output sekvensielt
- Bruker maskert self-attention
- Brukes i: GPT, Claude, Gemini

### Full encoder-decoder:

- Oversettelse, oppsummering
- T5, BART

### Nøkkelkomponenter:

- ① Multi-head self-attention
- ② Feed-forward nettverk
- ③ Layer normalization
- ④ Residual connections
- ⑤ Positional encoding

GPT = “Decoder-only”

Store språkmodeller (LLM) bruker typisk kun decoder-delen for tekstgenerering.

## G04: Forklare hva tokens er og hvordan tokenisering fungerer

**Tokens = tekstens “atomer”**

- LLM-er leser ikke bokstaver eller ord – de leser **tokens**
- Token  $\approx$  delord, ca. 4 tegn per token i gjennomsnitt

**Tokenisering – eksempel:**

“Pasienten har diabetes”  $\rightarrow$  [“Pas”, “ient”, “en”, “ har”, “ diab”, “etes”]

**Byte Pair Encoding (BPE):**

- Vanlig tokeniseringsmetode
- Bygger vokabular iterativt
- Balanserer mellom tegn og ord
- GPT-4: ca. 100 000 tokens

**Viktig å vite:**

- Lange ord = flere tokens
- Sjeldne ord splittes mer
- Påvirker kostnad og hastighet
- tiktoken (OpenAI) for telling

**Hvorfor tokens?**

Mer effektivt enn bokstaver (for kort) eller hele ord (vokabular for stort).

## G05: Forstå konseptet kontekstvindu (context window)

Kontekstvindu = modellens “arbeidsminne”

- Maksimalt antall tokens modellen kan “se” samtidig
- Inkluderer både input (prompt) **og** output

Eksempler på kontekstvinduer:

Modell	Kontekst	≈ Sider tekst
GPT-3.5	4K–16K	5–20 sider
GPT-4	8K–128K	10–150 sider
Claude 3	200K	~250 sider
Gemini 1.5 Pro	1M+	Hele bøker!

### Begrensninger

- For lange tekster: Må kuttes eller deles opp
- Modellen “glemmer” innhold utenfor vinduet

## G06: Forklare hva temperature betyr i tekstgenerering

**Temperature = kontroll over “kreativitet”**

**Teknisk:**

- Skalerer logits før softmax:  $p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$
- Påvirker sannsynlighetsfordelingen over neste token

**Lav temperature (0.0–0.3):**

- Mer deterministisk
- Velger mest sannsynlige tokens
- Konsistent, “trygt”
- ✓ Medisinsk info, fakta

**Høy temperature (0.7–1.0+):**

- Mer tilfeldig/kreativ
- Flere overraskende valg
- Mer variasjon
- ✓ Kreativ skriving, brainstorming

**Anbefaling for medisin**

Bruk **lav temperature (0.0–0.3)** for faktabaserte oppgaver. Høyere temperatur øker risiko for hallusinering.

## G07: Anvende zero-shot prompting

### Zero-shot = ingen eksempler

- Be modellen løse oppgaven **uten å vise eksempler**
- Baserer seg på modellens eksisterende kunnskap

### Eksempel:

#### Zero-shot prompt

Klassifiser følgende symptom som akutt eller kronisk:  
“Pasienten har hatt hodepine i 3 måneder.”

#### Når fungerer zero-shot?

- Modellen har sett lignende oppgaver i treningen
- Oppgaven er tydelig og veldefinert
- Domenet er godt representert i treningsdata

#### Begrensninger

For spesialiserte oppgaver eller uvanlige formater kan zero-shot gi dårlige resultater. Da trengs<sup>9/17</sup>

## G08: Anvende few-shot prompting

Few-shot = noen eksempler

- Gi modellen **2–5 eksempler** på ønsket input-output
- Modellen lærer mønsteret “in-context”

### Few-shot prompt

Klassifiser symptomvarighet:

Symptom: ‘Brystsmerter i 30 minutter’ → Akutt

Symptom: ‘Ryggsmarter i 2 år’ → Kronisk

Symptom: ‘Feber siden i går’ → Akutt

--

Symptom: ‘Hodepine i 3 måneder’ →

### Tips for effektiv few-shot:

- Velg **representative** eksempler
- Vis **variasjonen** i mulige outputs
- Hold eksemplene **konsistente** i format
- Start med 2-5 eksempler, juster ved behov

## G09: Anvende Chain-of-Thought (CoT) prompting

### Chain-of-Thought = vis resonnementet

- Be modellen **tenke steg-for-steg**
- Forbedrer ytelse på komplekse resonneringsoppgaver

#### Uten CoT

Bør pasienten henvises?

→ “Ja”

#### Med CoT

Tenk steg-for-steg:

1. Symptomer: X, Y, Z
  2. Alvorlighetsgrad: Moderat
  3. Risikofaktorer: Ja, alder
- Konklusjon: Bør henvises

### Varianter:

- **Zero-shot CoT:** “Let's think step by step”
- **Few-shot CoT:** Vis eksempler med resonnement
- **Self-consistency:** Generer flere CoT, velg majoritetsvar

## G10: Beskrive god praksis for systemprompts

**Systemprompt = instruksjon som setter kontekst**

**Elementer i et godt systemprompt:**

- ① **Rolle:** "Du er en medisinsk assistent..."
- ② **Kontekst:** "Brukeren er helsepersonell..."
- ③ **Oppførsel:** "Svar konsist og presist..."
- ④ **Begrensninger:** "Ikke gi diagnoser..."
- ⑤ **Format:** "Svar på norsk, bruk punktlister..."

**Eksempel på systemprompt**

Du er en medisinsk AI-assistent for helsepersonell. Svar på norsk. Vær presis og evidensbasert. Henvis til kilder når mulig. Gi aldri definitive diagnoser - foreslå differensialdiagnoser. Ved usikkerhet, si fra.

**Tips**

Test og iterer på systemprompts! Små endringer kan gi stor effekt.

## G11: Definere hallusinering og dens implikasjoner

**Hallusinering = modellen “finner på” feil informasjon**

- LLM-er genererer tekst som *høres* riktig ut, men er *faktisk feil*
- Ikke uvitenhet – modellen presenterer usannheter med overbevisning

**Typer hallusinasjoner:**

- **Faktiske feil:** Feil statistikk, datoer, navn
- **Oppdiktede kilder:** Referanser til artikler som ikke eksisterer
- **Logiske feil:** Inkonsistente resonnementer
- **Kontekstfeil:** Misforståelse av spørsmålet

**Implikasjoner i medisin**

- Feil dosering eller kontraindikasjoner kan være livstruende
- **Alltid verifiser** LLM-output mot pålitelige kilder!
- LLM = beslutningsstøtte, **ikke** erstatning for fagkunnskap

## G12: Kjenne til GPT-4, Claude, Gemini og deres anvendelser

Modell	Utvikler	Styrker	Anvendelse
GPT-4/4o	OpenAI	Bred kunnskap, multimedial	ChatGPT, API, Copilot
Claude 3	Anthropic	Langt kontekstvindu, "trygg"	Dokumentanalyse
Gemini	Google	Integrt i Colab, gratis	Kodehjelp, søk
Llama 3	Meta	Open source, kan kjøres lokalt	Forskning, tilpasning

### Medisinsk anvendelse:

- **Journaloppsummering** – Trekk ut nøkkelinformasjon
- **Litteratursøk** – Finn relevante artikler
- **Differensialdiagnostikk** – Støtte til klinisk resonnement
- **Pasientkommunikasjon** – Forklaringer på pasientnivå

## G13: Forklare konseptet grunnmodell (foundation model)

**Foundation model = pretrent på enorme datamengder**

- Trent på terabytes av tekst/bilder uten spesifikk oppgave
- Kan tilpasses (*finetuned*) til mange forskjellige oppgaver
- Koster millioner å trenne fra scratch

Karakteristikker:

- ① **Stor skala:** Milliarder av parametre
- ② **Selvveiledet læring:** Ingen labels nødvendig
- ③ **Emergente egenskaper:** Evner som “dukker opp” ved skala
- ④ **Tilpasningsbar:** Finetuning, prompting, RAG

Eksempler på foundation models

**Tekst:** GPT-4, Claude, Llama | **Bilder:** CLIP, DALL-E, Stable Diffusion | **Multimodal:** GPT-4V, Gemini Pro Vision

# G14: Beskrive RAG (Retrieval-Augmented Generation)

RAG = koble LLM til ekstern kunnskapsbase

Problemet RAG løser:

- LLM har “frossen” kunnskap fra treningstidspunkt
- Kan hallusinere fakta
- Mangler domene-spesifikk informasjon

Hvordan RAG fungerer:

- ① **Spørring:** Bruker stiller spørsmål
- ② **Retrieval:** Hent relevante dokumenter fra database
- ③ **Augmentation:** Legg dokumenter til prompt
- ④ **Generation:** LLM svarer basert på hentet kontekst

## Medisinsk anvendelse

- Koble LLM til oppdaterte retningslinjer, Felleskatalogen, UpToDate
- Svare basert på pasientens journal (med nødvendige tilganger)

# Oppsummering: Generativ AI og LLM

## Grunnleggende:

- G01: Generativ vs. diskriminativ AI
- G02–G03: Self-attention og transformer-arkitektur
- G04–G06: Tokens, kontekstvindu, temperature

## Prompt engineering:

- G07–G09: Zero-shot, few-shot, Chain-of-Thought
- G10: Systemprompts for konsistent oppførsel

## Utfordringer og avansert:

- G11: Hallusinering – kritisk i medisinsk kontekst
- G12–G14: Modeller, foundation models, RAG

## Lab 3 – Praktisk erfaring

Eksperimenter med prompting-teknikker, analyser tokenisering, diskuter etikk og begrensninger ved LLM i medisin.