

# Generativ AI og Store Språkmodeller

## ELMED219: Momentliste G01–G14

ELMED219

Vår 2026

- 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  $y$  gitt input  $x$
- 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  $x \rightarrow$  Kategori/Label  $y$     |    **Generativ:** Prompt  $\rightarrow$  Nytt innhold

### Medisinsk anvendelse

Generativ AI: Oppsummere journaler, generere rapporter, svare på spørsmål, syntetisere treningsdata

# 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”)

### 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** (svar)

### Eksempler på kontekstvinduer:

Modell	Kontekst	≈ Sider
GPT-4o	128K	~150 sider
GPT-5	256K+	~300 sider
Claude Opus 4.5	200K	~250 sider
Gemini 3 Pro	2M+	Hele bøker!

### Input vs. output:

- **Input-kontekst:** Hvor mye modellen kan lese
- **Output-kontekst:** Hvor langt svar den kan gi
- Output ofte mindre (4K–32K tokens)

### Begrensninger

For lange tekster må kuttes. Modellen “glemmer” utenfor vinduet. Større vindu = dyrere/tregere.

# G06: Forklare hva **temperatur** betyr i tekstgenerering

**Temperatur = 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 temperatur (0.0–0.3):**

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

**Høy temperatur (0.7–1.0+):**

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

Anbefaling for medisin

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

# G07: Anvende zero-shot prompting

**Zero-shot = ingen eksempler** – be modellen løse oppgaven **uten å vise eksempler**

**Eksempler på zero-shot prompts:**

## Zero-shot prompts

Klassifiser symptom som akutt/kronisk:

“Hodepine i 3 måneder.”

Oppsummer denne journalnotaten i 3 setninger.

Oversett til pasientvennlig språk:

“Bilateral pneumoni med hypoksemi.”

## Begrensninger

For spesialiserte oppgaver kan zero-shot gi dårlige resultater. Da trengs **few-shot** eksempler.

**Når fungerer zero-shot?**

- Modellen har sett lignende oppgaver
- Oppgaven er tydelig definert

# 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: ‘Ryggsmerter 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 3–5 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  
→ Konklusjon: Bør henvises

### Varianter:

- **Zero-shot CoT:** “Let’s think step by step”
- **Few-shot CoT:** Vis eksempler med resonnement

#### Innebygd CoT i nyere LLM-er

ChatGPT o1/o3 (Thinking), Claude Opus 4.5 (Extended Thinking), Gemini 3 (Deep Think) har CoT innebygd.

# 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
- **Oppdiktete kilder:** Referanser som ikke eksisterer
- **Logiske feil:** Inkonsistente resonnementer

## Utvikling:

- Nyere modeller hallusinerer *mindre*
- Kan *ikke* elimineres helt
- RAG og CoT reduserer risiko

## Implikasjoner i medisin

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

## G12: Kjenne til GPT-5, Claude, Gemini og deres anvendelser

Modell	Utvikler	Styrker	Anvendelse
GPT-5/o3	OpenAI	Multimodal, reasoning	ChatGPT, Copilot
Claude 4	Anthropic	200K kontekst, "trygg"	Dokumentanalyse
Gemini 3	Google	Integrert i Colab	Kodehjelp, søk
Llama 4	Meta	Open source, lokalt	Forskning

### Medisinsk anvendelse:

- **Journaloppsummering** – Nøkkelinformasjon
- **Litteratursøk** – Relevante artikler
- **Differensialdiagnostikk** – Klinisk støtte

### Lokale modeller:

- **Ollama, LM Studio**
- Destillering, kvantisering
- GDPR-vennlig, offline

[chat.uib.no](https://chat.uib.no)

UiB har egen AI-tjeneste med GDPR-hensyn for studenter og ansatte.

## G13: Forklare konseptet grunnmodell (foundation model)

**Foundation model = pretraint på enorme datamengder**

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

**Karakteristikk:**

- 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 (2025)

Tekst: GPT-5, Claude 4, Llama 4 | Bilder: DALL-E 3, Stable Diffusion 3 | Multimodal: Gemini 3, Sora

# G14: Beskrive RAG (Retrieval-Augmented Generation)

**RAG = koble LLM til ekstern kunnskapsbase**

## Problemet RAG løser:

- LLM har “frossen” kunnskap
- Kan hallusinere fakta
- Mangler domene-spesifikk info

## Hvordan RAG fungerer:

- 1 **Spørring:** Bruker stiller spørsmål
- 2 **Retrieval:** Hent dokumenter
- 3 **Augmentation:** Legg til prompt
- 4 **Generation:** LLM svarer

## Medisinsk anvendelse

- Koble LLM til retningslinjer, [Felleskatalogen](#), [UpToDate](#)
- Svare basert på pasientjournal (med tilganger)
- Reduserer hallusinerer ved å forankre svar i kilder

# Oppsummering: Generativ AI og LLM

## Grunnleggende:

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

## Prompt engineering:

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

## Utfordringer og avansert:

- G11: Hallusinerer – 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.