

# Generativ AI og Store Språkmodeller

## ELMED219: Momentliste G01–G14

ELMED219

Vår 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  $y$  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, Flan-T5

### 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 (for engelsk/norsk)

## Tokenisering – eksempel:

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

## BPE og språkforskjeller:

- Bygger vokabular iterativt
- GPT-4: ca. 100 000 tokens
- **Kinesisk:** Hvert tegn  $\approx$  1–2 tokens
- Forenklet/tradisjonell: Ulike tokens

## Viktig å vite:

- Lange/sjeldne ord = flere tokens
- Påvirker kostnad og hastighet
- **tiktoken** (OpenAI) for telling
- Ulike modeller = ulike tokenizers

## 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:  $p_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}}$
- **Effekt:** Lav  $T \rightarrow$  skarpere fordeling (høyeste logit dominerer); Høy  $T \rightarrow$  flatere fordeling (mer tilfeldighet)

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 skriving, brainstorming

Anbefaling for medisin

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

# G07: Anvende zero-shot prompting

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

**Eksempler på zero-shot prompts:**

## Zero-shot prompts

- 1 Klassifiser symptom som akutt/kronisk: "Hodepine i 3 mnd"
- 2 Oppsummer journalnotatet i 3 setninger.
- 3 Oversett til pasientvennlig språk: "Bilateral pneumoni"
- 4 Ekstraher alle medikamenter fra denne teksten.
- 5 Er denne lab-verdien unormal? Hb 8.2 g/dL

**Når fungerer zero-shot?**

- Modellen har sett lignende oppgaver
- Oppgaven er tydelig definert

## Begrensninger

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

# 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 differensialdiagnosar. 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*
- Også kalt *konfabulering* – begge termer brukes i litteraturen

## Typer hallusinasjoner:

- **Faktiske feil:** Feil statistikk, datoer, navn
- **Oppdiktede 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	Integrt i Colab	Kodehjelp, søk
Llama 4	Meta	Open source, lokalt	Forskning

## Lokale modeller:

- Ollama, LM Studio
- Destillering: Liten modell lærer fra stor
- Kvantisering: Redusert presisjon ( $16 \rightarrow 4$  bit)
- GDPR-vennlig, offline

chat.uib.no

Styrker: GDPR-ok, ingen datalagring, gratis for UiB

Svakheter: Eldre modell, begrenset kontekst, ikke multimodal

# 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 (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:

- ➊ **Spørring:** Bruker stiller spørsmål
- ➋ **Retrieval:** Hent dokumenter
- ➌ **Augmentation:** Legg til prompt
- ➍ **Generation:** LLM svarer

Medisinsk anvendelse

- Koble LLM til retningslinjer, [Felleskatalogen](#), [UpToDate](#)
- Svare basert på pasientjournal (med tilganger)
- Reduserer hallusinering ved å forankre svar i kilder
- Klinisk beslutningsstøtte med oppdatert evidens
- Søk i institusjons-spesifikke prosedyrer og protokoller

# 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: 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.