

Lokale en cloud-AI in webontwikkeling: ecologische impact, kosten en een beslissingskader voor duurzame integratie.

Alexandru Poenaru.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Mevr. F. Spriet

Co-promotor: Dhr. A. Sorella

Academiejaar: 2025–2026

Eerste examenperiode

Departement IT en Digitale Innovatie .

**HO
GENT**

Woord vooraf

Toen Turtle Srl, een bedrijf met een uitgesproken focus op duurzaamheid, ESG en een groenere toekomst, mij deze onderzoeksvraag voorlegde, sprak het thema mij onmiddellijk aan. Artificiële intelligentie is vandaag onlosmakelijk verweven met ons dagelijks leven, en de maatschappij staat voor de uitdaging om op economisch, politiek en sociaal vlak een weloverwogen koers te bepalen in dit nieuwe tijdperk. Wat daarbij te vaak naar de achtergrond verdwijnt, zeker naarmate modellen krachtiger en geavanceerder worden, is de ecologische voetafdruk die deze technologie nalaat. Dit onderzoek biedt de kans om net dat aspect te belichten, en ik vind dit een bijzonder waardevolle bijdrage aan het discours rond duurzame AI-integratie. Graag maak ik van dit voorwoord gebruik om een aantal mensen uitdrukkelijk te bedanken.

In de eerste plaats wil ik mijn medestudenten bedanken die samen met mij stage hebben gelopen bij Turtle Srl: Arnaud Tison, Tom Kluskens, Kyandro Voet en Bavo Vancoppennolle. De voortdurende uitwisseling van bevindingen, vragen en perspectieven heeft mijn denkkader aanzienlijk verruimd en was een onmisbare meerwaarde voor dit onderzoek. Ik kijk met veel voldoening terug op onze samenwerking.

In de tweede plaats gaat mijn oprechte dank uit naar mijn promotor, Fien Spriet, voor haar voortdurende inzet, waardevolle feedback en scherpe inzichten doorheen het volledige onderzoeksproces.

Daarnaast wil ik mijn co-promotor, Antonio Sorella, en Turtle Srl van harte bedanken voor de begeleiding bij de afbakening van de scope, de gerichte sturing en het ter beschikking stellen van alle nodige middelen om dit onderzoek tot een goed einde te brengen.

Een bijzonder woord van dank gaat ook naar mijn ouders, Marinela Crismac en Costel-Viorel Crismac, die mij gedurende mijn volledige studieloopbaan onvoorwaardelijk hebben gesteund, niet alleen met raad en daad, maar bovenal met een rotsvast vertrouwen dat mij keer op keer de kracht gaf om door te zetten.

Samenvatting

De toenemende integratie van Large Language Models (LLM's) in webapplicaties stelt organisaties voor een ecologisch dilemma: cloud providers bieden schaalbaarheid maar zijn ondoorzichtig qua milieu-impact, terwijl lokale hosting controle biedt maar hardware-investeringen vereist. Kleine en middelgrote ondernemingen (KMO's) erkennen de meerwaarde van LLM's voor het automatiseren van bedrijfsprocessen, maar missen de empirische inzichten om de ecologische en economische implicaties van hun infrastructuurkeuzes objectief af te wegen. De casus van Turtle Srl, een Italiaans duurzaamheidsinnovator, maakt dit urgent: een infrastructuur voor artificiële intelligentie (AI) met een onverantwoorde ecologische voetafdruk ondermijnt hun eigen kernmissie.

De centrale onderzoeksvraag luidt: hoe beïnvloedt de integratie van LLM's in webontwikkelingsframeworks de ecologische voetafdruk in vergelijking met gecentraliseerde generatieve AI-diensten? Het onderzoek beoogde de energie- en CO₂-uitstoot van beide architecturen empirisch te meten voor het toepassingsscenario *Document Information Extraction*, en de bevindingen te vertalen naar een praktisch beslissingskader voor KMO's, IT-architecten en webontwikkelaars.

Daartoe werden 6241 inferentieaanroepen uitgevoerd over zes modellen: 3 cloud-varianten via de HuggingFace Hub en 3 lokale varianten (in 4-bit kwantisatie), op Italiaanse energiefacturen. Energie werd gemeten met CodeCarbon en EcoLogits. Omdat EcoLogits de prefill-fase standaard negeert, wat bij een input/output-verhouding van $\approx 150:1$ een onderschatting met factor ≈ 31 veroorzaakt, werd een empirisch bepaalde correctiefactor κ toegepast.

Cloud-inferentie blijkt per 1.000 tokens 3,7 tot 8,5 keer energie-efficiënter dan lokale inferentie (gemiddeld $\approx 5\times$), met een CO₂-uitstoot van $\approx 5,8$ versus $\approx 21,6$ mgCO₂eq/1.000 tokens in de Italiaanse context, en is bovendien 2,3× sneller. Op het vlak van extractiekwaliteit scoort Llama 3.1 8B het best (62–66/100 op ground-truth vergelijking), terwijl Apertus 8B lokaal dramatisch faalt (14,9/100), vermoedelijk doordat de Q4_K_M-kwantisatie de niet-standaard architectuurcomponenten van Apertus (xIELU-activatiefunctie, QK-Norm) suboptimaal ondersteunt. Voor het goedkoopste cloud-model is lokale hardware bij realistisch KMO-volume financieel nooit rendabel; voor duurdere modellen ligt het break-even punt bij $\approx 2,6$ miljoen aanroepen.

De resultaten kennen inherente beperkingen: CodeCarbon heeft een meetafwijking van $\approx 40\%$, de cloud-energieschatting draagt onzekerheid via de correctiefactor, en de studie is beperkt tot één toepassingsscenario en hardwareconfiguratie.

Desondanks bieden de bevindingen een direct bruikbare leidraad voor het werkveld: een beslissingsboom begeleidt IT-architecten op basis van 9 variabelen. Cloud verdient de ecologische en economische voorkeur tenzij privacy een harde vereiste is of hoge volumes de hardware-investering amortiseren.

Inhoudsopgave

Lijst van figuren	x
Lijst van tabellen	xii
Lijst van codefragmenten	xiii
Lijst van afkortingen	xiv
1 Inleiding	1
1.1 Probleemstelling	2
1.2 Onderzoeksvraag	2
1.3 Onderzoeksdoelstelling	3
1.4 Opzet van deze bachelorproef	4
2 Stand van zaken	5
2.1 Duurzaamheid en ecologische impact van LLM's	5
2.2 Lokale versus cloudgebaseerde inferentie en offloading	7
2.2.1 Theorie kwantisatie	10
2.3 Meetmethoden voor energie-, CO ₂ - en waterimpact	10
2.4 Locatie, elektriciteitsmix en infrastructuur	12
2.5 Architecturale en softwarematige tactieken voor groene AI-systemen	14
2.6 Efficiëntie, kosten en kwaliteitsafwegingen in webtoepassingen	15
2.7 Transparantie, regulering en governance rond duurzame AI	17
2.8 Synthese en positionering van deze bachelorproef	19
3 Methodologie	21
3.1 Fase 1: Literatuur en casusverkenning (week 1–2)	21
3.2 Fase 2: Requirementsanalyse en experimenteel ontwerp (week 3–4)	21
3.3 Fase 3: Implementatie van de testomgeving (week 5–6)	22
3.4 Fase 4: Simulatie en dataverzameling (week 7–9)	22
3.5 Fase 5: Analyse en interpretatie (week 10–11)	23
3.6 Fase 6: Beslissingskader en rapportering (week 12)	23
3.7 Beperkingen van het Onderzoek	23
4 Componentenselectie: Modellen en Meetinstrumenten	26
4.1 Componentenselectie: Longlist en Shortlist	26
4.1.1 Selectie van instrumenten voor energie- en CO ₂ -metingen	26
4.1.2 Empirische Normalisatie: Berekening van de Lokale PUE	28

4.1.3	Selectie van cloud- en lokale LLM's	29
5	Requirementsanalyse en Experimenteel Ontwerp	34
5.1	Requirementsanalyse en Bedrijfscontext	34
5.2	Systeemgrenzen en Meetbereik	35
5.2.1	Definitie van het lokale meetbereik (On-Premise)	35
5.2.2	Definitie van het cloud-meetbereik (API)	35
5.2.3	Normalisatie van de Werklast (Prefill vs. Decoding)	36
5.2.4	Kwantisatie en Functionele Pariteit.	38
5.2.5	Definitie van Metrics en KPI's.	38
5.2.6	Experimenteel Protocol en Tool-Configuratie	41
6	Implementatie van de Testomgeving	43
6.1	Documentextractie en Voorverwerking	43
6.2	Datamodellering en Semantische Sturing via Pydantic.	45
6.2.1	Schemaontwerp en Extractieregels.	45
6.2.2	Grammaticagestuurde decodering.	45
6.3	Implementatie van de Energie- en Emissiemetingen	46
6.3.1	Directe Hardware-instrumentatie met CodeCarbon.	46
6.3.2	PUE-correctie en Regionale Emissiefactoren	46
6.3.3	Cloud-estimatie via EcoLogits	48
7	Simulatie en Dataverzameling	54
7.1	Orkestratie en Serverlevenscyclus	54
7.1.1	Cloud Inferentie: Document-eerst batchverwerking	54
7.1.2	Lokale Inferentie: Serverlevenscyclus en Batching	55
7.2	Resultatenopslag	55
7.3	Evaluatie en Scoring	56
8	Resultaten / Analyse en interpretatie van data	57
8.1	Energieverbruik	58
8.1.1	Energie-efficiëntie per 1000 tokens	58
8.1.2	Uitsplitsing lokale energie: GPU, CPU en RAM.	59
8.1.3	Spreiding van het energieverbruik	60
8.1.4	Relatie tussen documentlengte en energieverbruik.	61
8.2	Prefill-correctiefactor en impact op cloud-energie	62
8.3	Energie-verhouding per modelkoppel	64
8.4	CO ₂ -uitstoot	64
8.4.1	CO ₂ -efficiëntie per 1000 tokens	65
8.4.2	Regionale gevoeligheidsanalyse: effect van datacenterlocatie.	65
8.4.3	CO ₂ -efficiëntie per regio	68

8.5	Latentie en doorvoer	69
8.5.1	Totale inferentietijd	69
8.5.2	Time To First Token (TTFT)	70
8.5.3	Doorvoer (tokens per seconde).	71
8.5.4	Relatie tokenvolume en inferentietijd	72
8.6	Extractiekwaliteit	73
8.6.1	Successrate	73
8.6.2	Structurele validatiescore	74
8.6.3	Inhoudsnauwkeurigheid (ground-truth vergelijking)	75
8.7	Kosten en break-even analyse.	77
8.7.1	Variabele kosten per 1 miljoen tokens	77
8.7.2	Break-even analyse: alle modelkoppels	78
8.7.3	Kosten per succesvolle extractie	80
8.8	Statistische toetsen.	80
8.9	Synthese van de resultaten	81
9	Beslissingskader voor duurzame AI-integratie	83
9.1	Van empirische data naar beslissingsvariabelen.	83
9.2	Belastingsprofiel als proxy voor het toepassingsscenario	84
9.3	Het beslissingskader	86
9.4	Bespreking van de beslissingsknopen	88
9.4.1	Knoop G1 – Dataprivacy.	88
9.4.2	Knoop G2 – GPU-hardwarebeschikbaarheid	88
9.4.3	Knoop G3 – Dagelijks aanroepvolume	88
9.4.4	Knoop G4 – Gemiddelde inputlengte per aanroep.	88
9.4.5	Knoop G5 – Cloud-API-tarief	89
9.4.6	Knoop G6 – Real-time latentievereiste	89
9.4.7	Knoop G7 – Break-even bereikbaarheid	89
9.4.8	Knoop G8 – Lokale stroommix	90
9.4.9	Knoop G9 – Kwaliteitsvalidatie van het lokale model	90
9.5	Toepassing op de casus Turtle Srl	91
9.6	Generaliseerbaarheid en beperkingen van het kader	92
10	Conclusie	93
10.1	Antwoord op de onderzoeksvragen	94
10.1.1	Deelvragen in het probleemdomein	94
10.1.2	Deelvragen in het oplossingsdomein	95
10.2	Kritische reflectie en beperkingen	96
10.3	Toekomstig onderzoek	97
A	Onderzoeksvoorstel	99
A.1	Samenvatting	99
A.2	Inleiding	100

A.3	Literatuurstudie	102
A.3.1	Duurzaamheid en AI-impact	102
A.3.2	Lokale versus cloudgebaseerde inferentie	102
A.3.3	Energieverbruik per token en emissie-intensiteit.	103
A.3.4	Implementatie- en infrastructuurimpact	103
A.3.5	Meetinstrumenten voor energie en CO ₂	103
A.3.6	Samenvatting	104
A.4	Methodologie	104
A.5	Verwacht resultaat, conclusie	106

Bibliografie**108**

Lijst van figuren

2.1	Gemiddelde CO ₂ -intensiteit elektriciteitsmix België en Italië.	12
8.1	Energie-efficiëntie per model (mWh per 1 000 tokens).	58
8.2	Energieuitsplitsing lokale modellen per 1 M tokens (GPU/CPU/RAM). . .	59
8.3	Spreiding energie-efficiëntie per model (violinplot, mWh per 1 000 tokens).	60
8.4	Inputtokenlengte versus energieverbruik per aanroep (lokale modellen, scatter).	61
8.5	Verdeling van de EcoLogits-prefill-correctiefactor κ (cloud-aanroepen). . .	62
8.6	Cloud-energie: onbewerkt EcoLogits versus prefill-gecorrigeerde bovengrens.	63
8.7	Energieverhouding lokaal/cloud per modelkoppel ($\times 3,7$ – $\times 8,5$).	64
8.8	CO ₂ -efficiëntie per model (gCO ₂ eq per 1 000 tokens).	65
8.9	Cloud CO ₂ -uitstoot per 1 M tokens naar datacenterlocatie.	66
8.10	Cloud CO ₂ per datacenterregio versus lokale referentie Italië.	67
8.11	CO ₂ -efficiëntie per cloud-regio (tokens per gCO ₂ eq).	68
8.12	Gemiddelde totale inferentietijd per aanroep per model ($\pm 1\sigma$).	69
8.13	Gemiddelde Time-To-First-Token (tijd tot het eerste gegenereerde token) (TTFT) per model ($\pm 1\sigma$).	70
8.14	Gemiddelde doorvoer per model: prefill-fase (links) en decode-fase (rechts).	71
8.15	Totaal tokenvolume versus inferentietijd per aanroep (cloud links, lokaal rechts).	72
8.16	Successrate (correct geparseerde JSON-uitvoer) per model.	73
8.17	Gemiddelde structurele validatiescore per model (succesvolle extracties).	74
8.18	Ground-truth nauwkeurigheidsscore per model en per-veld heatmap. . .	75
8.19	Variabele kosten per 1 M tokens: cloud API versus lokale elektriciteit. . .	77
8.20	Cumulatieve kostenvergelijking cloud API versus lokaal per modelkoppel.	78
8.21	Energieverbruik per succesvolle extractie (gecorrigeerd voor faalpercentage).	80
9.1	Beslissingsboom voor de infrastructuurkeuze van Large Language Model (grootschalig taalmodel) (LLM)-integratie.	87

A.1 Planning bachelorproef 2025–2026	106
--	-----

Lijst van tabellen

4.1	Hardwareconfiguratie van de lokale inferentie-opstelling bij Turtle Srl.	30
4.2	Requirements-matrix voor de modelselectie. (• = requirement aanwezig; - = requirement niet aanwezig)	32
4.3	Definitieve modelselectie en vergelijkingsparen.	32
5.1	API-tarieven per model in USD en EUR (per 1M tokens)	40
6.1	Gebruikte Python-bibliotheken in het Proof of Concept (PoC)-framework	44
6.2	Geëvalueerde lokale modellen (GGML Unified Format (modelbestandsformaat voor lokale inferentie) (GGUF)-formaat)	46
6.3	llama-server configuratieparameters (lokale inferentie)	47
6.4	CodeCarbon- en inferentieparameters (lokale inferentie)	48
6.5	Regionale emissiefactoren (kgCO ₂ eq/kWh) gebruikt in het PoC. Gegevens uit Ember (2026)	48
6.6	Geëvalueerde cloud-modellen (HuggingFace Hub)	50
6.7	EcoLogits- en inferentieparameters (cloud-inferentie)	50
8.1	Gemiddelde variabele kost per aanroep (basis voor de break-even berekening). Hardware €1.400 is vaste kost en niet inbegrepen in de bovenstaande waarden.	79
8.2	Resultaten van de Welch-t-toetsen voor de vergelijking cloud vs. lokaal.	81
9.1	Beslissingsvariabelen voor de infrastructuurkeuze.	84
9.2	Typische AI-toepassingscenario's gekarakteriseerd via belastingsprofiel.	86

Lijst van codefragmenten

4.1	Illustratie van de integratie van CodeCarbon via de Python contextmanager, met de llama-server HTTP API als inferentie-backend.	27
6.1	Het definitieve Pydantic-extractieschema (<code>modello.py</code>): <code>FactuurModel</code> combineert JSON-schemavalidatie met Italiaanse semantische ankerpunten in de docstring.	51
6.2	Isolatie en registratie van hardwarematig energieverbruik via CodeCarbon in <code>LlamaCppProvider</code>	52
6.3	Manuele registratie van aangepaste modellen in de <code>EcoLogits-modeldatabank</code>	53
7.1	Per-aanroep isolatie via <code>LocalServerManager</code>	56

Lijst van afkortingen

AI Artificial Intelligence.

API Application Programming Interface.

AR/VR Augmented Reality / Virtual Reality.

BEP Break-Even Point.

CapEx Capital Expenditure (investeringsuitgaven).

CO₂ koolstofdioxide.

CO₂eq CO₂-equivalent.

CPU Central Processing Unit.

DIMM Dual Inline Memory Module.

ETS Emissions Trading System.

FLOPS Floating Point Operations Per Second.

FP16 16-bit Floating Point (halfprecisie zwevende-kommanotatie).

GBNF GGML Backus-Naur Form (grammatica-formaat voor constrained decoding in llama.cpp).

GEMV General Matrix-Vector Multiplication.

GGUF GGML Unified Format (modelbestandsformaat voor lokale inferentie).

GHG Greenhouse Gas.

GPU Graphics Processing Unit.

HF HuggingFace.

ICT Information and Communication Technology.

IoT Internet of Things.

JSON JavaScript Object Notation.

KMO Kleine of Middelgrote Onderneming.

KPI Key Performance Indicator.

KV Key-Value.

LCA Life Cycle Assessment.

LLM Large Language Model (grootschalig taalmodel).

LoRA Low-Rank Adaptation (parameter efficiënte fine-tuning techniek).

NIM NVIDIA Inference Microservices.

NLP Natural Language Processing (natuurlijke-taalverwerking).

NVML NVIDIA Management Library.

OCR Optical Character Recognition.

OpEx Operational Expenditure.

PCIe Peripheral Component Interconnect Express.

PoC Proof of Concept.

PSU Power Supply Unit (computervoeding).

PUE Power Usage Effectiveness (verhouding totaal datacenterverbruik / IT-verbruik).

RAG Retrieval-Augmented Generation.

RAM Random Access Memory.

RAPL Running Average Power Limit.

TDP Thermal Design Power (maximaal warmtevermogen van een processor).

TTFT Time-To-First-Token (tijd tot het eerste gegenereerde token).

VRAM Video Random Access Memory.

1

Inleiding

De hedendaagse digitale transitie wordt in toenemende mate gekenmerkt door de integratie van generatieve **Artificial Intelligence (AI)** binnen webapplicaties. Waar webontwikkeling zich voorheen voornamelijk richtte op deterministische datastromen, vereist de moderne applicatieontwikkeling een verschuiving naar dynamische, semantisch gedreven interacties. Deze technologische evolutie wordt vandaag de dag mogelijk gemaakt door de opkomst van **Large Language Models (LLM's)** (Singh e.a., 2024). Deze modellen bieden software-ingenieurs ongeziene mogelijkheden voor het automatiseren van complexe, cognitieve taken.

Binnen de context van bedrijfsapplicaties is er met name een sterke vraag naar systemen die ongestructureerde documentatie accuraat omzetten naar gestructureerde informatie. De specifieke focus van dit onderzoek ligt op het toepassingsscenario *Document Information Extraction*. Bedrijven verwerken dagelijks immense hoeveelheden documenten, zoals complexe duurzaamheidsrapporten en industriële stuklijsten. Generatieve taalmodellen bezitten het architecturale vermogen om de onderliggende betekenis te abstraheren en de geëxtraheerde informatie in een gestandaardiseerd formaat te retourneren.

Hoewel de voordelen van dergelijke **AI**-integraties evident zijn, brengt de verschuiving naar **AI**-gedreven webapplicaties een aanzienlijke ecologische kost met zich mee (Morrison e.a., 2025). Recente literatuur toont aan dat de continue emissies tijdens de operationele inferentiefase de eenmalige uitstoot van de oorspronkelijke training ruimschoots kunnen overstijgen bij intensief gebruik (Singh e.a., 2024).

Hier ontstaat een urgent dilemma voor applicatieontwikkelaars: de keuze tussen gecentraliseerde cloud-**Application Programming Interfaces (API's)** en het lokaal hosten van modellen op eigen infrastructuur. Cloud-oplossingen bieden directe schaalbaarheid, maar het emissieprofiel is ondoorzichtig. Lokale architecturen bieden controle en de mogelijkheid tot het afstemmen van de hardware op de exacte

werklust, maar vereisen initiële investeringen en introduceren ecologische uitdagingen rondom de productie van componenten en stand-by verliezen (Clemm e.a., 2024; Morrison e.a., 2025). Dit onderzoek, dat experimenteel zal worden uitgevoerd op een on-premise infrastructuur in Italië, streeft ernaar deze afweging inzichtelijk te maken voor ontwikkelaars.

1.1. Probleemstelling

De versnelde integratie van generatieve AI in webapplicaties heeft geleid tot een buitengewoon complex vraagstuk voor **Kleine of Middelgrote Ondernemingen (KMO's)**. Zij erkennen de meerwaarde van taalmodellen voor het stroomlijnen van applicaties en het automatiseren van bedrijfsprocessen. Het ontbreekt hun echter aan de empirische inzichten en kaders om de ecologische en economische implicaties van hun infrastructuurkeuzes objectief af te wegen (Krupp e.a., 2025). De primaire doelgroep van deze bachelorproef bestaat uit webontwikkelaars, technische applicatiearchitecten en -managers binnen deze softwarebedrijven, die acute behoefte hebben aan vertaalbare, datagedreven richtlijnen.

De urgentie van dit probleem wordt perfect geïllustreerd door de casus van Turtle Srl, de opdrachtgever en co-promotor van dit onderzoeksproject. De missie van Turtle Srl is gedreven door de huidige ecologische en sociale uitdagingen; zij begeleiden bedrijven en instellingen in hun transitie naar duurzaamheid. Dit doen zij enerzijds door het berekenen van objectieve duurzaamheids-**Key Performance Indicators (KPI's)** via hun ViVACE-model, en anderzijds door het plannen van transitie waarbij technologische oplossingen worden geïntegreerd om de milieu-, economische en sociale impact te minimaliseren (Turtle-Srl, 2026). De kracht van het bedrijf ligt in de samenwerking tussen academische instellingen en de industrie, waarbij transparantie en objectieve data centraal staan.

Wanneer Turtle Srl AI-functionnalité wenst te integreren voor de geautomatiseerde extractie van documentinformatie uit duurzaamheidsrapporten, stuiten zij op een fundamentele paradox. Indien de onderliggende AI-infrastructuur een onverantwoorde ecologische voetafdruk heeft, ondermijnt dit hun eigen kernmissie. Het technische team van Turtle Srl overweegt zowel cloud-API's als on-premise hosting in Italië, maar mist een wetenschappelijk onderbouwd kader om te bepalen welke specifieke infrastructuurkeuze de laagste ecologische voetafdruk combineert met de onmisbare kostenefficiëntie.

1.2. Onderzoeksvraag

Dit onderzoek vertrekt vanuit één overkoepelende hoofdonderzoeksvraag, die verder wordt gespecificeerd in gerichte deelvragen binnen zowel het probleemdomain als het oplossingsdomein.

De hoofdonderzoeksvraag luidt:

Hoe beïnvloedt de integratie van LLM's in webontwikkelingsframeworks de ecologische voetafdruk in vergelijking met het gebruik van gecentraliseerde generatieve AI-diensten?

Deelvragen in het probleemdomain:

- Wat is het energieverbruik en de koolstofdioxide (CO₂)-uitstoot van cloudgebaseerde AI-aanroepen in een webapplicatie? En die van lokaal gehoste AI-modellen?
- Hangt de keuze af van het toepassingsscenario binnen de applicatie? Of is de ene optie beter in alle scenario's?
- Welke methoden zijn geschikt om de impact te meten van deze AI-modellen?

Deelvragen in het oplossingsdomain:

- Hoeveel energie en CO₂-uitstoot genereren beide systemen (lokaal en cloud) bij het verwerken van eenzelfde AI-werklast?
- Hoe verhouden performantie en energie-efficiëntie zich tussen een lokaal model en een cloud-LLM binnen een webomgeving?
- Kunnen de resultaten vertaald worden naar praktische richtlijnen voor duurzame AI-integratie in webontwikkeling?

1.3. Onderzoeksdoelstelling

De onderzoeksdoelstelling formuleert het beoogde resultaat van deze bachelorproef en splitst zich op in twee tastbare opleveringen.

Het eerste doel is de ontwikkeling en implementatie van een Proof of Concept (PoC) webomgeving. Deze architectuur dient als gecontroleerde testomgeving waarin de specifieke werklast van *Document Information Extraction* gelijktijdig via twee paden wordt uitgevoerd: enerzijds via netwerkconnecties naar gecentraliseerde cloud-modellen en anderzijds direct naar lokaal gehoste modellen op de on-premise infrastructuur in Italië. Het criterium voor succes is het accuraat en reproduceerbaar registreren van de instrumentatiemetrieken, waarbij het fysieke hardware-energieverbruik en de netwerk-overhead succesvol worden omgezet in vergelijkbare CO₂-equivalents (CO₂-equivalenten).

Het tweede beoogde resultaat is de synthese van deze experimenteel verzamelde data in een praktisch en direct inzetbaar beslissingskader voor de doelgroep. Dit kader, vormgegeven als een beslissingsboom, moet IT-managers en webontwikkelaars stapsgewijs begeleiden bij hun architecturale keuze. Succes is bereikt wanneer dit kader eenduidig kan aangeven onder welke specifieke contextuele parameters (zoals documentvolume, responstijd en stroommix) een lokale integratie de cloud-integratie ecologisch en economisch overtreft.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de verschillende fasen van het onderzoek beschreven.

In Hoofdstuk 4 worden de meetinstrumenten geselecteerd en de longlist en shortlist van modellen opgesteld.

In Hoofdstuk 5 worden de functionele en niet-functionele requirements vastgesteld en uitgewerkt in een formeel experimenteel protocol, inclusief de definitie van de KPI's en de systeemgrenzen.

In Hoofdstuk 6 wordt de implementatie van de testomgeving en de meetinfrastructuur beschreven.

In Hoofdstuk 7 worden de gesimuleerde werklasten uitgevoerd, de dataverzameling beschreven en de extractieresultaten gescored ten opzichte van de ground truth en de structuurkwaliteit.

In Hoofdstuk 8 worden de verzamelde data en empirische resultaten uit de testomgeving gepresenteerd en kritisch geanalyseerd.

In Hoofdstuk 9 worden de analytische bevindingen vertaald naar een praktisch beslissingskader voor IT-architecten en webontwikkelaars. Dit kader, vormgegeven als een beslissingsboom, stuurt de infrastructuurkeuze stapsgewijs op basis van negen contextparameters: dataprivacy, hardwarebeschikbaarheid, aanroepvolume, inputtokenlengte, cloud-tarief, latentievereiste, economische Break-Even Point (BEP), stroommix en kwaliteitsvalidatie.

In Hoofdstuk 10, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

In dit hoofdstuk wordt de bachelorproef gepositioneerd binnen de bredere literatuur over duurzaamheid van grootschalige taalmodellen (LLM's), hun energie- en materiaalvoetafdruk en de concrete implicaties voor webtoepassingen die generatieve AI integreren. Aansluitend op de inleiding wordt eerst de ecologische impact van LLM's in kaart gebracht, vervolgens het spanningsveld tussen lokale en cloud-gebaseerde inferentie besproken, en ten slotte worden meetmethoden, locatie-gebonden factoren, architecturale optimalisaties, beleidskaders en performantie- en kostenafwegingen behandeld. Het hoofdstuk vormt zo de theoretische basis voor de methodologie en de casestudie bij Turtle Srl in de volgende hoofdstukken, waarin de in de literatuur geschetste afwegingen experimenteel worden onderzocht.

Daarnaast besteedt de literatuur aandacht aan minder zichtbare aspecten van duurzaamheid, zoals waterverbruik in grootschalige datacenters en de rol van datacenterarchitectuur en werklustplanning. In deze bachelorproef wordt waterverbruik echter niet afzonderlijk gemeten voor de lokale opstelling; de kwantitatieve analyse focust op elektriciteitsverbruik en daaruit afgeleide CO₂-uitstoot, terwijl waterverbruik enkel op hoog niveau wordt besproken in de context van cloud- en datacentergebaseerde scenario's. Dit sluit aan bij de vaststelling in de literatuur dat gereedschappen voor watermetingen en optimalisatie nog vooral in onderzoeksomgevingen voorkomen (Y. Jiang e.a., 2025; Shumba e.a., 2024), wat in een praktische bachelorproef met beperkte middelen een realistische afbakening rechtvaardigt.

2.1. Duurzaamheid en ecologische impact van LLM's

De snelle opmars van LLM's heeft geleid tot een sterke toename in rekenverbruik, energieconsumptie en bijhorende CO₂-uitstoot, zowel tijdens training als tijdens

inferentie (Singh e.a., 2024). Recente overzichtsstudies benadrukken dat de milieu-impact niet langer als neveneffect mag worden beschouwd, maar als een volwaardige ontwerpconstraint naast accuraatheid, latency en kosten, omdat het groot-schalige gebruik in datacenters significante emissies genereert (Singh e.a., 2024). Daarbij wordt gewezen op de verschuiving van de impact van de trainingsfase naar de inferentiefase zodra modellen op grote schaal in productie worden ingezet, waardoor dagelijkse gebruikspatronen dominant worden in de totale voetafdruk (Fu e.a., 2024; Singh e.a., 2024). Projecties gaan er zelfs vanuit dat datacenters tegen 2030 mogelijk meer dan 11,7% van de totale Amerikaanse energievraag voor hun rekening nemen (Morrison e.a., 2025), wat aantoont dat de sector zich in een fase van meer dan evenredige groei bevindt die de capaciteitsplanning van energienetwerken onder druk zet.

Om de omvang van die impact concreet te maken, documenteren Morrison e.a. (2025) als eersten de volledige milieuvoetafdruk van het ontwikkeltraject van een reeks taalmodellen, van 20 miljoen tot 13 miljard parameters, getraind op maximaal 5,6 biljoen tokens. De bevindingen zijn opvallend: wanneer hardwareproductie, modelontwikkeling en de uiteindelijke trainingsruns samen worden meegeteld, bedraagt de totale CO₂-uitstoot bijna 500 metrische ton, vergelijkbaar met het jaarlijkse elektriciteitsverbruik van circa honderd gemiddelde Amerikaanse huishoudens (Morrison e.a., 2025). Daarnaast werd ruim 2,7 miljoen liter water verbruikt bij koelingsprocessen, zelfs in een datacenter dat als uitzonderlijk waterefficiënt wordt beschouwd. Een bijkomende en verrassende bevinding is dat de modelontwikkelingsfase (het iteratieve tunen van hyperparameters en het uitvoeren van voorbereidende experimenten vóór de definitieve training) alleen al goed was voor circa 50% van de emissies van de definitieve trainingsrun. Dit aandeel wordt door vrijwel geen enkele modelontwikkelaar publiek gerapporteerd (Morrison e.a., 2025).

Een ander onderschat fenomeen is de instabiliteit van het vermogensprofiel tijdens training. Morrison e.a. (2025) stellen vast dat het stroomverbruik gedurende een trainingsrun niet constant is, maar fluctueert tussen circa 15% en 85% van het maximale vermogen van de gebruikte hardware. Dergelijke grote schommelingen bemoeilijken niet alleen accurate emissieberekeningen op basis van gemiddeld verbruik, maar vormen ook een uitdaging voor de stabiliteit van het elektriciteitsnet bij een verdere opschaling van AI-trainingsinfrastructuur.

Een holistische benadering van de milieu-impact maakt een onderscheid tussen operationele emissies (stroomverbruik tijdens training en inferentie) en de zogenoemde *embodied carbon* van hardware, netwerkinfrastructuur en koeling (Morrison e.a., 2025; Wu e.a., 2021). Studies over duurzame AI pleiten ervoor om de volledige levenscyclus van modellen, data en infrastructuur te beschouwen, inclusief hardwareproductie, vervanging en end-of-life, omdat anders belangrijke verschuivingen in de klimaatimpact onzichtbaar blijven (Clemm e.a., 2024; Morrison e.a., 2025). Dit is bijzonder relevant voor lokale opstellingen: als servers grotendeels on-

derbenut zijn of snel verouderen, kan de gepercipieerde winst tegenover gedeelde cloudinfrastructuur afnemen. Wu e.a. (2021) illustreren dit vanuit industrieel perspectief: bij Facebook werden meerdere optimalisatiecycli doorgevoerd waarbij het operationele energieverbruik gemiddeld 20% per zes maanden daalde, maar waarbij tegelijk de *embodied carbon* van de hardwaregeneraties mee in rekening werd gebracht.

Een belangrijk nuancepunt betreft de verhouding tussen de trainingsfase en de inferentiefase in de totale koolstofvoetafdruk. Terwijl de training van een groot model al snel enkele honderden tonnen CO₂ kost, tonen studies aan dat voor taalmodellen die op grote schaal worden ingezet, de inferentiefase op termijn de dominante kostenfactor wordt (Singh e.a., 2024; Wu e.a., 2021). Dit staat in contrast met, bijvoorbeeld, aanbevelingssystemen, waarbij training en inferentie ruwweg even zwaar doorwegen (Wu e.a., 2021). De verklaring is dat taalmodellen bij elk verzoek een relatief grote hoeveelheid rekenkracht vereisen, en dat productiesystemen doorgaans miljoenen verzoeken per dag verwerken. Voor een bedrijf zoals Turtle Srl betekent dit dat zelfs een beperkt dagelijks gebruik van een cloud-API over een langere periode een substantieel cumulatieve voetafdruk kan opbouwen.

Naast de koolstofvoetafdruk komt in de literatuur ook de watervoetafdruk van AI in beeld, met name voor grootschalige trainings- en inferentieclusters in datacenters (Li e.a., 2023; Morrison e.a., 2025). Onderzoek naar de “verborgen” waterconsumptie van dergelijke installaties toont aan dat vooral koelingssystemen en waterintensieve elektriciteitsproductie leiden tot aanzienlijke waterverbruikscijfers bij training en inferentie (Li e.a., 2023).

Recente surveys over efficiënte en resource-bewuste LLM's wijzen op de sterke variatie in energieverbruik tussen verschillende modelarchitecturen, parameteromvang en optimalisatietechnieken (G. Bai e.a., 2024; Wan e.a., 2023). Technieken zoals modelcompressie, kwantisatie, distillatie en sparsity kunnen de benodigde *Floating Point Operations Per Second (FLOPS)* en geheugentoegang aanzienlijk reduceren, maar gaan soms gepaard met kwaliteitsverlies op complexe taken; het ecologische winstpotentieel moet daarom telkens afgewogen worden tegen de functionele eisen van de toepassing (Wan e.a., 2023; Xu e.a., 2024). Deze literatuur leidt tot een eerste kernvraag voor de bachelorproef: welke modelconfiguraties zijn “goed genoeg” voor de casus van Turtle Srl en leveren tegelijk een substantieel milieuwinstpotentieel op?

2.2. Lokale versus cloudgebaseerde inferentie en offloading

Binnen de literatuur wordt steeds vaker een onderscheid gemaakt tussen inferentie op lokaal beheerde infrastructuur (on-premise of on-device) en inferentie via gecentraliseerde cloud-LLM-diensten (G. Bai e.a., 2024; Singh e.a., 2024). Zoals ver-

meld in Sectie 2.1, kunnen optimalisatietechnieken zoals modelcompressie, kwantisatie en distillatie de benodigde rekenkracht per inferentieverzoek drastisch reduceren, waardoor lokale inzet van kleinere, taakgespecialiseerde modellen haalbaar wordt met aanzienlijk lager energieverbruik per verzoek (G. Bai e.a., 2024; Wan e.a., 2023). Empirisch onderzoek naar inferentie-energie bevestigt dat de juiste toepassing van optimalisaties het energieverbruik met tot 73% kan verlagen ten opzichte van niet-geoptimaliseerde basislijnen, maar dat de effectiviteit sterk afhangt van hardware, softwareframework en de vorm van de werklast (Fernandez e.a., 2025). Voor KMO's is lokale inferentie echter niet vanzelfsprekend, omdat investeringskosten, beheercomplexiteit en benodigde expertise reële drempels vormen (Singh e.a., 2024).

Een belangrijk inzicht uit recent empirisch onderzoek is dat inferentie zelf geen uniform energieprofiel heeft. Fernandez e.a. (2025) maken onderscheid tussen de *prefill*-fase, waarbij de inputprompt in parallel verwerkt wordt, en de *decoding*-fase, waarbij de output token voor token serieel gegenereerd wordt. Beide fasen hebben een ander vermogensprofiel en worden anders beïnvloed door batchgrootte en hardware-architectuur. Statische batching (waarbij de batchgrootte gedurende de hele inferentie constant gehouden wordt) leidt daardoor tot **Graphics Processing Unit (GPU)**-onderbenutting wanneer de outputlengtes sterk variëren: zodra kortere sequenties klaar zijn, wacht de GPU inactief op de langere. Dit betekent concreet dat de effectiviteit van een gegeven optimalisatie sterk afhankelijk is van het *belastingsprofiel*: de verhouding tussen input- en outputlengte en de variabiliteit daarin (Fernandez e.a., 2025). Voor een toepassing zoals documentextractie, waarbij inputprompts lang maar de outputs relatief kort en gestructureerd zijn, levert dit een wezenlijk andere energiebalans op dan bij een open-end chatbot (Fernandez e.a., 2025).

Cloudproviders bieden daarentegen schaalbare diensten met zeer grote, breed toepasbare modellen, die goed scoren op brede prestatietests, maar mogelijk minder efficiënt zijn voor een afgebakende bedrijfstoepassing en vaak minder transparant zijn over hun energie- en emissieprofiel (Sapkota e.a., 2025; Singh e.a., 2024). De literatuur benadrukt dat de keuze tussen lokaal en cloud niet enkel een technische, maar ook een economische en governancebeslissing is, onder meer door afhankelijkheid van leveranciers, datalocatie, beveiligingsvereisten en compliance met sectorale regelgeving (Hacker, 2023).

Eén van de eerste systematische empirische studies naar de energie van LLM-inferentie op echte hardware is het werk van Samsi e.a. (2023), die het LLaMA-model in varianten van 7B tot 65B parameters evalueerden op zowel NVIDIA V100- als A100-GPU's. Een opvallende bevinding is dat modernere GPU-architecturen een aanzienlijk betere energie-efficiëntie bieden: de A100 presteerde met name voor kleinere modellen duidelijk beter dan de V100, terwijl het energieprofiel bij grotere modellen sterk samenhangt met de noodzaak tot model sharding over meerdere

GPU's. Bovendien toonden de experimenten aan dat de complexiteit van de invoertaak het energieverbruik meetbaar beïnvloedt: wiskundige redeneertaken vereisen andere verwerkingspatronen dan eenvoudigere instructiefollowtaken (Samsi e.a., 2023). Dit onderstreept dat een enkelvoudige energie-per-token-schatting een te grove vereenvoudiging is, en dat de inputkarakteristieken altijd mee in rekening moeten worden gebracht.

Naast energetische en economische overwegingen speelt ook *data privacy* een centrale rol bij de architectuurkeuze. Bij cloudproviders verlaten documenten de eigen infrastructuur en worden verwerkt op servers van derden, wat in industriële contexten in conflict kan komen met vertrouwelijkheidsverplichtingen of sectorale regelgeving (Hacker, 2023; Singh e.a., 2024). Lokale inferentie elimineert dit risico volledig, omdat data en model op de eigen hardware blijven. Voor een bedrijf zoals Turtle Srl, dat werkt met bedrijfsgevoelige duurzaamheidsrapportage van klanten, is dit een niet-functionele vereiste die minstens even zwaar weegt als de ecologische afweging, en die in de praktijk de keuze naar een lokale of hybride architectuur kan sturen.

Een tussenpositie wordt ingenomen door frameworks voor *local-cloud offloading*, waarbij inferentieverzoeken dynamisch worden verdeeld over lokale en cloudinfrastructuur op basis van factoren zoals taakcomplexiteit, beschikbare rekencapaciteit en latency-eisen (Kim e.a., 2023; Wang e.a., 2025). Dergelijke hybride benaderingen laten toe eenvoudige of privacygevoelige taken lokaal af te handelen, terwijl zwaardere of sporadische werklasten naar de cloud worden verplaatst, wat zowel performantie- als energievoordelen kan opleveren afhankelijk van de context (G. Bai e.a., 2024; Wang e.a., 2025). Onderzoek binnen draadloze LLM-services illustreert bovendien dat een geïntegreerde benadering, die ook de netwerkcomponent meeneemt, bijkomende optimalisaties toelaat, bijvoorbeeld door inferentie dichterbij de gebruiker te brengen of requests te clusteren (Wang e.a., 2025).

Voor mobiele en edge-scenario's worden federated benaderingen onderzocht, waarbij de verfijning van LLM's naar gebruikersapparaten wordt verschoven om de privacygevoeligheid van trainingsdata te beschermen; zo verwerkt elk apparaat uitsluitend zijn eigen lokale data zonder die te centraliseren (Xu e.a., 2023). Hoewel dit domein zich vooral richt op smartphones en Internet of Things (IoT), zijn de principes ook relevant voor webapplicaties die LLM-functionaliteit in de browser of nabijgelegen edge-servers willen integreren, bijvoorbeeld om netwerkverkeer en centrale rekenlast te verminderen (Xu e.a., 2024). Dit roept de vraag op in welke mate de casus van Turtle Srl baat kan hebben bij een dergelijke "edge-achtige" architectuur, of dat een klassiek client-servermodel met een centrale backend volstaat.

Een bijkomende strategie om grotere modellen op beperkte consumenten-GPU's te draaien is *Central Processing Unit (CPU)/GPU-offloading*, waarbij een deel van de modellagen in systeem-Random Access Memory (RAM) wordt geplaatst en

door de CPU wordt verwerkt, terwijl de overige lagen op de GPU blijven (Song e.a., 2023). Hoewel dit in theorie toelaat modellen van 13B tot 30B parameters te laden op een GPU met slechts 8 GB Video Random Access Memory (VRAM), introduceert elke laagovergang een overdracht van activatiedata via de Peripheral Component Interconnect Express (PCIe)-bus, wat bij toenemend offloaden een groeiende bandbreedte-bottleneck veroorzaakt en de generatiesnelheid significant doet dalen (Gerganov & contributors, 2023). Vanuit duurzaamheidsperspectief is dit relevant omdat de langere inferentietijd leidt tot een substantieel hoger energieverbruik per token in vergelijking met volledig GPU-geladen inferentie, zelfs wanneer het instantane systeemvermogen vergelijkbaar blijft (Samsi e.a., 2023). Kwantisatie blijkt daarbij de meest doeltreffende optimalisatie: door de precisie te verlagen van 16-bit Floating Point (halfprecisie zwevende-kommanotatie) (FP16) naar 4-bit kan een model dat anders niet in het VRAM past volledig op de GPU worden geladen, waardoor de offloading-boete geheel wordt vermeden (Khan e.a., 2025).

2.2.1. Theorie kwantisatie

In essentie is kwantisatie een compressietechniek waarbij de numerieke precisie van de modelparameters (de gewichten) wordt gereduceerd, bijvoorbeeld van 16-bit floating point (FP16) naar 4-bit of 5-bit integers. Dit verkleint de geheugen-voetafdruk van het model aanzienlijk (vaak met een factor drie tot vier), waardoor grotere modellen op consumenten-GPU's met beperkt VRAM zoals de RTX 3070 kunnen draaien. Hoewel dit proces gepaard kan gaan met een lichte daling in nauwkeurigheid (de zogenaamde *quantization error*), tonen recente methoden zoals GGML Unified Format (modelbestandsformaat voor lokale inferentie) (GGUF) aan dat dit verlies bij 4-bit of 5-bit kwantisatie vaak verwaarloosbaar is voor de meeste praktische toepassingen, terwijl de winst in inferentiesnelheid en energie-efficiëntie substantieel is (Khan e.a., 2025; Wan e.a., 2023).

2.3. Meetmethoden voor energie-, CO₂- en waterimpact

Om gefundeerde keuzes te maken tussen lokale en cloudgebaseerde inferentie is een betrouwbare meting of schatting van energieverbruik en CO₂-uitstoot essentieel (Morrison e.a., 2025). In lokale omgevingen wordt hiervoor vaak gebruikgemaakt van softwarebibliotheken die metingen van CPU-, GPU- en geheugenbelasting koppelen aan landspecifieke emissiefactoren, zoals CodeCarbon of gelijkaardige tools, zodat onderzoekers tijdens experimenten rechtstreeks een schatting in kilogram CO₂-equivalent bekomen (Milutinović & Lukić, 2025). Deze tools integreren typisch in Python-omgevingen en zijn daardoor goed inzetbaar in prototypische webbackends die LLM-aanroepen orkestreren; in deze bachelorproef worden zij gebruikt om het elektriciteitsverbruik en de CO₂-uitstoot van de lokale opstelling te benaderen.

Naast runtimebibliotheken bestaan er generieke rekentools zoals Green Algorithms, die op basis van hardwareconfiguratie, looptijd en datacenterkenmerken de koolstofvoetafdruk van arbitraire berekeningen inschatten (Lannelongue e.a., 2021). Vergelijkende studies van verschillende meetinstrumenten tonen echter aanzienlijke verschillen in scope, nauwkeurigheid en aannames, bijvoorbeeld rond de efficiëntie van het datacenter of de **Power Usage Effectiveness (verhouding totaal datacenterverbruik / IT-verbruik) (PUE)** (Bannour e.a., 2021). Hierdoor is transparante rapportering van de gebruikte tool, parameters en emissiefactoren noodzakelijk om resultaten tussen studies te kunnen vergelijken (Bannour e.a., 2021; Morrison e.a., 2025).

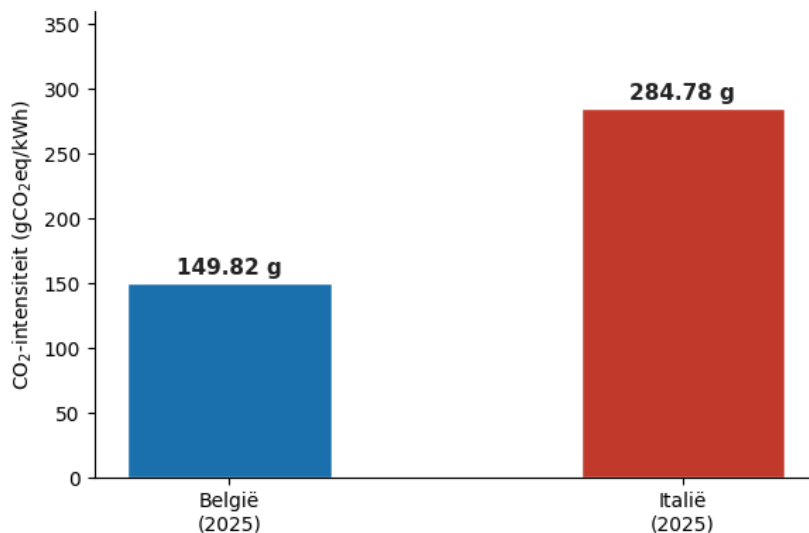
Voor cloudcontext is directe meting van energieverbruik per verzoek meestal niet beschikbaar voor eindgebruikers; onderzoekers zijn daarom aangewezen op indirecte proxies zoals het aantal verwerkte tokens, het type onderliggende hardware en algemene energie-per-token-schattingen uit de literatuur (Fu e.a., 2024; Singh e.a., 2024). Recent werk rond LLMCO₂ toont dat de variabiliteit in batchgrootte, promptlengte en tokenoutput, gecombineerd met hardware- en implementatieverschillen, het moeilijk maakt om eenvoudige formule-gebaseerde schattingen nauwkeurig te houden, en stelt meer geavanceerde voorspellingsmodellen voor (Fu e.a., 2024). Dit ondersteunt de keuze om in deze bachelorproef conservatieve en duidelijk gedocumenteerde aannames te hanteren bij het omrekenen van tokenstatistieken naar energie- en emissieschattingen.

Voor de meting van cloud-inferentie worden in de literatuur twee fundamenteel verschillende benaderingen onderscheiden. De eerste is een *top-down* benadering, waarbij het energieverbruik wordt afgeleid uit het aantal verwerkte tokens gecombineerd met een gemiddeld energie-per-token-getal op basis van gepubliceerde hardwarespecificaties of empirische studies (Fu e.a., 2024; Samsi e.a., 2023). De tweede is een *bottom-up* levenscyclusanalyse (**Life Cycle Assessment (LCA)**), zoals toegepast door EcoLogits: hierbij worden niet alleen de elektriciteitsvraag van de inferentie zelf, maar ook de hardwareproductie-emissies en de **PUE** van het clouddatacenter mee in rekening gebracht (Berthelot e.a., 2024; Rincé & Banse, 2025b). Het voordeel van de **LCA**-benadering is dat ze een vollediger beeld geeft van de daadwerkelijke milieu-impact over de hele levenscyclus; het nadeel is dat ze sterk afhankelijk is van aannames over het onderliggende hardwarepark en de serverlocatie, die cloudproviders zelden publiek maken. Beide benaderingen zijn daardoor inherent schattingen met een zekere onzekerheid, wat transparante rapportering van aannames onmogelijk maakt (Bannour e.a., 2021; Morrison e.a., 2025). Naast koolstofemissies komt ook de kwantificatie van waterverbruik en andere milieu-indicatoren op (Li e.a., 2023). Tools en datasets zoals WaterWise en regionale water-efficiëntiedatasets voor datacenters maken het mogelijk om job-scheduling en locatiekeuzes te optimaliseren op zowel koolstof- als watervoetafdruk, al bevinden deze zich nog hoofdzakelijk in onderzoeksomgevingen (Y. Jiang e.a., 2025;

Shumba e.a., 2024). In de context van een bachelorproef is het doorgaans niet haalbaar om alle deze dimensies volledig door te meten, maar de literatuur suggereert dat het zinvol is om ten minste de afhankelijkheid van emissiefactoren per land expliciet te maken en, waar mogelijk, ook watergerelateerde implicaties te bespreken (Lannelongue e.a., 2021; Li e.a., 2023).

2.4. Locatie, elektriciteitsmix en infrastructuur

De klimaatimpact van eenzelfde hoeveelheid verbruikte elektrische energie kan sterk variëren naargelang de herkomst en de samenstelling van de elektriciteitsmix (Lannelongue e.a., 2021). Databronnen zoals Ember (2026) tonen aan dat de gemiddelde emissiefactor per kWh in Italië hoger ligt dan in landen met een groter aandeel kernenergie en hernieuwbare bronnen, zoals België, waardoor identieke rekenwerklasten in een Italiaans datacenter meer CO₂-uitstoot genereren dan in een Belgisch (zie Figuur 2.1). Voor deze bachelorproef is dit bijzonder relevant omdat de proof-of-concept infrastructuur in Italië draait, terwijl België als referentiescenario wordt beschouwd.



Figuur 2.1: Gemiddelde CO₂-intensiteit van de elektriciteitsmix in België (149.82 gCO₂eq/kWh, jaargemiddelde 2025) en Italië (284.78 gCO₂eq/kWh, jaargemiddelde 2025) (Ember, 2026).

Recente systemen voor *koolstofbewuste* en *waterbewuste* taakplanning gaan nog een stap verder door rekentaken in de tijd en ruimte te verplaatsen naar periodes en locaties met een lagere koolstof- of waterintensiteit (Y. Jiang e.a., 2025; Kim e.a., 2023; Lee e.a., 2024). Een concreet voorbeeld hiervan is het GreenScale-framework, dat voor intensieve taken zoals AI-inferentie of Augmented Reality / Virtual Reality (AR/VR)-rendering dynamisch kiest tussen lokale verwerking op een mobiel apparaat, een nabijgelegen edge-server of een centrale cloud-datacenter. Wanneer de lokale stroommix op dat moment sterk afhankelijk is van fossiele brandstoffen, kan het systeem besluiten de taak te verplaatsen naar een datacenter in een regio waar

op dat moment een overschot aan zonne- of windenergie is. Hierbij wordt niet enkel gekeken naar het actuele verbruik, maar wordt ook de 'koolstofafschrijving' van de gebruikte hardware meegerekend, waardoor een ouder, minder efficiënt apparaat soms vermeden wordt ondanks een gunstige lokale energiemix (Kim e.a., 2023). Studies over grootschalige computingsystemen suggereren dat dergelijke koolstof- en waterbewuste planning, gecombineerd met efficiëntere hardware en koeling, substantiële emissiereducties kan opleveren zonder dat het prestatieniveau drastisch hoeft te dalen (Lee e.a., 2024; Qi e.a., 2023). Deze inzichten tonen aan dat niet alleen de keuze tussen lokaal en cloud van belang is, maar ook de interne organisatie van werklasten binnen een gekozen infrastructuur, een inzicht dat in het beslissingskader van deze bachelorproef mee wordt verwerkt.

Een aanvullend maar minder zichtbaar aspect is de temporele variatie van de koolstofintensiteit. Niet alleen verschilt de emissiefactor per regio, maar ook binnen eenzelfde land fluctueert die factor sterk naargelang het tijdstip van de dag en het seizoen: op momenten dat zonne- of windenergie een overschot genereert, daalt de marginale emissiefactor aanzienlijk, terwijl piekvraag doorgaans wordt opgevangen door emissie-intensievere gascentrales (Kim e.a., 2023; Lee e.a., 2024). Systemen zoals Carbon Connect en GreenScale exploiteren dit principe door rekentaken, voor zover de latency-eisen het toelaten, te verschuiven naar momenten en locaties met een gunstigere mix (Lee e.a., 2024). Voor een lokale opstelling in Italië, waar de marginale koolstofintensiteit sterk afhangt van de beschikbaarheid van zonne-energie overdag versus aardgas 's nachts, kan dit een bijkomend argument zijn om batchwerklasten te plannen op momenten met een lagere emissiefactor. Daarbovenop wijzen levenscyclusanalyses van AI-systemen erop dat ook de fysieke infrastructuur (serverhardware, netwerken, koeling) en de vervangingscycli bijdragen tot de totale emissie (Morrison e.a., 2025; Wu e.a., 2021). Wanneer lokale servers onderbenut blijven, kan de per-verzoek-voetafdruk onverwacht hoog uitvallen, terwijl gedeelde cloudinfrastructuur de embodied carbon over vele gebruikers spreidt maar de individuele zichtbaarheid van de impact verkleint (Clemm e.a., 2024; Morrison e.a., 2025). Dit werpt de vraag op in hoeverre een klein tot middelgroot bedrijf als Turtle Srl voldoende schaal en voorspelbaarheid heeft om lokale infrastructuur duurzaam te benutten.

De mate van hardware-benutting is een cruciale maar vaak onderschatte variabele bij de duurzaamheidsbeoordeling van lokale opstellingen. Wanneer een lokale server uitsluitend voor LLM-inferentie wordt ingezet en gedurende het grootste deel van de dag inactief is, wordt de embodied carbon van de hardware (de emissies die gepaard gingen met de productie van de GPU, het moederbord en de koeling) over een beperkt aantal verzoeken gespreid (Morrison e.a., 2025; Wu e.a., 2021). In dat scenario kan de per-verzoek-voetafdruk van de lokale opstelling hoger uitvallen dan die van een cloudplatform, dat dezelfde hardware over duizenden gelijktijdige gebruikers deelt. De break-even in embodied carbon ligt doorgaans pas bij een

minimum bezettingsgraad die voor veel KMO's moeilijk haalbaar is zonder aanvullende werklasten op dezelfde infrastructuur te draaien (Clemm e.a., 2024). Dit is een directe designconstraint voor de casus van Turtle Srl: de RTX 3070-opstelling is pas ecologisch verantwoord als ze voldoende intensief en consistent wordt benut. Ten slotte maakt de literatuur duidelijk dat geografische verschillen in klimaat en waterschaarste mee bepalen hoe zwaar de watervoetafdruk van AI-installaties doorweegt (Li e.a., 2023; Shumba e.a., 2024). In regio's met beperkte watervoorraden of waterintensieve stroommixen kan het wenselijk zijn om rekentaken te verplaatsen naar andere regio's, zelfs als dit betekent dat de CO₂-uitstoot licht toeneemt, wat een nieuw spanningsveld creëert tussen verschillende duurzaamheidsdoelstellingen (Y. Jiang e.a., 2025). Dit maakt duidelijk dat de vergelijking tussen Italië en België in deze bachelorproef idealiter niet alleen op CO₂, maar ook op bredere milieu-indicatoren wordt geïnterpreteerd, al ligt de nadruk in de casus primair op CO₂-uitstoot.

2.5. Architecturale en softwarematige tactieken voor groene AI-systemen

Naast keuzes op infrastructuurniveau beschrijft de literatuur verschillende architecturale en softwarematige tactieken om de koolstofvoetafdruk van ML- en LLM-gedreven systemen te reduceren (Järvenpää e.a., 2023). Een recente synthese van groene architecturale tactieken voor ML-enabled systemen groepeert maatregelen in categorieën zoals *workload reduction* (bijvoorbeeld caching, request-throttling, adaptieve sampling), *efficiency improvement* (bijvoorbeeld hardwareacceleratie, parallelisatie, energie-efficiënte algoritmes) en *carbon-intensity awareness* (bijvoorbeeld planning op basis van emissiefactoren) (Clemm e.a., 2024; Järvenpää e.a., 2023). Deze tactieken sluiten nauw aan bij ontwerpbeslissingen in webarchitecturen, zoals hoe vaak en wanneer LLM-aanroepen worden gedaan, hoe resultaten hergebruikt worden en welke delen van een workflow echt generatieve AI vereisen.

In de context van webapplicaties komen hier nog domeinspecifieke optimalisaties bij, zoals het beperken van contextlengte door betere pre- en postprocessing, gebruik van Retrieval-Augmented Generation (RAG) om generatieve modellen alleen in te zetten waar nodig, en het combineren van heuristieken of klassieke algoritmen met LLM's (Wan e.a., 2023). Dergelijke patronen kunnen het aantal tokens per verzoek, en dus het energieverbruik per taak, aanzienlijk verlagen zonder dat de gebruikerservaring daaronder lijdt (Fu e.a., 2024; Krupp e.a., 2025).

De impact van ontwerpbeslissingen op het energieverbruik wordt bijzonder scherp geïllustreerd door Krupp e.a. (2025), die twee webagents met fundamenteel verschillende architecturen vergeleken: LASER, een agent die gebruikmaakt van het grote gesloten GPT-4-model zonder preprocessing, en MindAct, een

agent die open-source LLM's combineert met slimme contextreductie. Uit hun schattingen blijkt dat LASER per voltooide webactie een energieverbruik van bijna 3 kWh genereert, wat de auteurs vertalen naar een koolstofequivalent dat vergelijkbaar is met tientallen kilometers autorijden. MindAct behaalt vergelijkbare taakresultaten met een fractie van dit verbruik; de energieverhouding tussen beide agents bedraagt circa drie orden van grootte (Krupp e.a., 2025). Bovendien toont het onderzoek aan dat hogere energieconsumptie niet automatisch leidt tot betere resultaten: efficiënte agents die slim gebruik maken van kleinere modellen voor een eerste filtering, presteren op sommige prestatietests even goed als de energieverblindende aanpak. Dit principe (een twee-staps aanpak waarbij een kleiner model eerst het aantal tokens reduceert dat een groter model moet verwerken) is direct toepasbaar op het documentextractie-toepassingsscenario van Turtle Srl. Bovendien suggereert de literatuur over groene architecturale tactieken dat niet-ML-specifieke maatregelen, zoals het optimaliseren van databasequeries, het reduceren van netwerkverkeer, en efficiënt beheer van deployment- en managementprocessen, eveneens een betekenisvolle bijdrage kunnen leveren aan de totale emissiereductie van een systeem (Järvenpää e.a., 2023; Wu e.a., 2021). Voor de casus van Turtle Srl betekent dit dat het beslissingskader niet enkel moet focus- sen op de keuze tussen lokaal en cloud, maar ook op concrete ontwerprichtlijnen voor de gehele webstack rond de LLM-integratie.

Een bijzonder toegankelijke optimalisatietactiek die geen herarchitecturering ver- eist, is gerichte promptoptimalisatie. Door de inputprompt beknopter te formule- ren, overbodige context weg te laten en de gewenste outputstructuur expliciet te specificeren, kan het aantal verwerkte tokens per verzoek aanzienlijk dalen (Fu e.a., 2024; Krupp e.a., 2025). Aangezien het energieverbruik van LLM-inferentie sterk ge- correleerd is met de totale tokenlengte (input én output), is dit een nulkostenop- timalisatie die geen extra hardware of modelwijzigingen vereist. Promptoptimali- satie vormt dus een eerste verdedigingslinie tegen onnodig hoog energieverbruik, ongeacht of het uiteindelijke model lokaal of via de cloud wordt aangesproken.

2.6. Efficiëntie, kosten en kwaliteitsafwegingen in webtoe- passingen

Voor webapplicaties die LLM-functionaliteit integreren, spelen naast duurzaam- heid ook responstijd, schaalbaarheid, beschikbaarheid en operationele kosten een centrale rol (Singh e.a., 2024). Surveys over efficiënte LLM's geven een over- zicht van optimalisatietechnieken op model- en systeemniveau, zoals kwantisatie (zie Sectie 2.2.1), parameterdeling, adapterlagen (bijv. Low-Rank Adaptation (parameterefficiënte fine-tuningtechniek) (LoRA)) en request-batching, die de inferencekosten per verzoek kunnen reduceren zonder noemenswaardige ach- teruitgang in antwoordkwaliteit voor veel bedrijfstoepassingen (G. Bai e.a., 2024;

Wan e.a., 2023). Hieruit volgt een spanningsveld tussen “maximale” modelprestaties en “voldoende goede” prestaties binnen strengere duurzaamheids- en budgetbeperkingen.

Recente studies naar de energie-efficiëntie van LLM's tonen aan dat kwantisatie (zoals INT4 of Q4_K_M) niet enkel de geheugenvoetafdruk verkleint, maar ook het operationele energieverbruik drastisch verlaagt. Onderzoek toont aan dat 4-bit kwantisatie, in vergelijking met standaard FP16-precisie, de opslagbehoefte met een factor vier reduceert en tegelijkertijd een significante energie-efficiëntiewinst oplevert tijdens matrixvermenigvuldigingen of General Matrix-Vector Multiplication (GEMV) (Khan e.a., 2025; Wan e.a., 2023). Dit impliceert dat een directe vergelijking tussen een lokaal gekwantiseerd model en een cloud-model op FP16-precisie geen zuivere hardware-vergelijking is, maar een vergelijking van architecturale implementaties.

Daarnaast is het energieprofiel van LLM-inferentie sterk asymmetrisch. Zoals uiteengezet in Sectie 2.2, hebben de *prefill*- en *decode*-fase een fundamenteel ander energieprofiel en worden zij anders beïnvloed door het belastingsprofiel (Fernandez e.a., 2025). Voor het toepassingsscenario van documentextractie — met lange inputprompts en compacte, gestructureerde outputs — is dit onderscheid bijzonder relevant: hoewel een inputtoken gemiddeld vijf à zes keer minder energie kost per token dan een outputtoken (Caravaca e.a., 2025), maakt de extreme input/output-verhouding (~150:1 bij de documenten in dit onderzoek) de prefill-fase alsnog tot de dominante energiepost. Een holistische meting die beide tokentypen met hun respectieve energiegewicht verrekent, is dan ook noodzakelijk voor een accuraat energiebeeld (Caravaca e.a., 2025; Fernandez e.a., 2025).

Tot slot introduceert de dwingende extractie naar gestructureerde formaten (zoals JavaScript Object Notation (JSON)) een eigen computationele overhead. Grammaticagestuurde decodering of *gestructureerde uitvoer* vereist actieve filtering van de token-probabiliteiten. Hoewel dit structurele validiteit garandeert, introduceert de grammatica-engine een “verborgen kost” in de vorm van extra CPU/GPU-cycli per gegenereerde token, wat de energie-efficiëntie per verzoek beïnvloedt (Dong e.a., 2024).

Op systeemniveau wordt gewezen op het belang van cachemechanismen, werkverdelingsstrategieën en architecturale keuzes zoals RAG (zie Sectie 2.5) om het aantal dure LLM-aanroepen te beperken (Wan e.a., 2023). Economische analyses tonen dat cloudgebaseerde LLM-API's vooral aantrekkelijk zijn bij lage of sterk fluctuerende werklast door het per-gebruik-model, terwijl lokale modellen bij hoog en voorspelbaar volume kostenefficiënter kunnen worden, zeker wanneer de hardware al beschikbaar is en energieprijzen en -mix gunstig zijn (G. Bai e.a., 2024; Singh e.a., 2024). Dit suggereert dat een belangrijk deel van het beslissingskader rond werklastkarakterisatie en volumescenario's moet draaien.

De economische break-even-analyse vormt een cruciaal onderdeel van de beslis-

singslogica tussen lokale en cloudgebaseerde inferentie. Bij cloudproviders zijn de kosten variabel: ze worden per verwerkte token aangerekend, wat bij lage of onregelmatige werklust een aantrekkelijk per-gebruik-model oplevert. Bij lokale inferentie zijn de relevante kosten grotendeels vast: de aanschaf en afschrijving van de GPU-hardware (de *Capital Expenditure (investeringsuitgaven) (CapEx)*), aangevuld met de operationele elektriciteitskosten per verzoek (de *Operational Expenditure (OpEx)*) (G. Bai e.a., 2024; Singh e.a., 2024). Het break-even punt (het volume waarbij de lokale totaal kost de cloudkost begint te onderschrijven) is sterk afhankelijk van de afschrijvingsperiode van de hardware, de elektriciteitsprijs ter plaatse en de cloud-API-tarieven voor het gekozen model. Uit Europese data blijkt dat de gemiddelde industriële elektriciteitsprijs in Italië hoger ligt dan het EU-gemiddelde, wat het lokale OpEx-voordeel ten opzichte van cloud gedeeltelijk uitholt (Eurostat, 2022). Dit break-even punt wordt in dit onderzoek empirisch bepaald op basis van de gemeten energie-per-verzoek-waarden en de actuele API-tarieven, en is één van de centrale KPI's van de experimentele analyse.

Ten slotte wordt in recente evaluatiestudies aangehaald dat de inhoudelijke kwaliteit van LLM-uitvoer (nauwkeurigheid, nuance en domeinspecifieke betrouwbaarheid) vaak achterblijft bij de oppervlakkige kwaliteit zoals vloeiendheid en coherentie. Bulian e.a. (2023) tonen aan dat linguïstische vloeiendheid niet correleert met feitelijke juistheid: modellen kunnen zeer overtuigend en autoritair klinken terwijl ze hallucinaties of onjuiste cijfers presenteren met een misleidende stelligheid. Dit gebrek aan onzekerheidsbesef is problematisch voor data-extractie, aangezien het model geen onderscheid maakt tussen wetenschappelijk onderbouwde feiten en statistisch gegenereerde onjuistheden. Hoewel dit inzicht afkomstig is uit onderzoek naar klimaatcommunicatie, is het principe breder toepasbaar: voor toepassingen rond duurzaamheidsrapportering, documentextractie en analyse van industriële stuklijsten is het beperken van hallucinaties en het correct omgaan met cijfermatige informatie eveneens belangrijk, wat kan pleiten voor kleinere, domeingegerichte modellen in combinatie met goede evaluatiemetriecken en testdatasets (Bulian e.a., 2023; Singh e.a., 2024). De literatuur laat echter open in welke mate dergelijke modellen in de praktijk voldoende zijn voor complexe industriële documenten, wat deze bachelorproef gedeeltelijk tracht te beantwoorden via de casus bij Turtle Srl.

2.7. Transparantie, regulering en governance rond duurzame AI

Naast technische en economische aspecten groeit de aandacht voor juridische en beleidsmatige kaders rond duurzame AI (Hacker, 2023; Singh e.a., 2024). Werk rond “sustainable AI regulation” benadrukt dat bestaande en toekomstige regelgeving, zoals de Europese AI Act en bredere milieuwetgeving, meer focus zou moeten leg-

gen op energie- en materiaalgebruik van AI-systemen, bijvoorbeeld via rapporteringsverplichtingen of duurzaamheids-by-design-principes (Hacker, 2023). Transparantie over de koolstof- en watervoetafdruk van AI-diensten wordt daarbij gezien als een noodzakelijke voorwaarde om zowel gebruikers als beleidsmakers in staat te stellen geïnformeerde keuzes te maken (Sapkota e.a., 2025).

Bij het analyseren van het concrete regelgevingskader valt op dat de huidige verplichtingen aanzienlijk beperkter zijn dan wat Hacker (2023) bepleit. Artikel 11 van de EU AI Act verplicht aanbieders van hoog-risico AI-systemen weliswaar om de gebruikte computationele middelen te documenteren, maar legt géén expliciete verplichting op om het energieverbruik te rapporteren; daarmee is de vergelijkbaarheid van de milieu-impact beperkt tot schattingen op basis van gedocumenteerde hardware (Hacker, 2023). Parallel hieraan introduceert de Energiebeheerrichtlijn (EU) 2023/1791 transparantieplichtingen voor datacenters, maar dit kader geldt niet voor kleinschalige on-premise opstellingen zoals die bij Turtle Srl. Opvallend is ook dat de Information and Communication Technology (ICT)-sector als geheel al verantwoordelijk is voor naar schatting 3,9% van de mondiale broeikasgasemissies (meer dan de wereldwijde luchtvaartsector met 2,5%), wat de urgentie van sectorspecifieke regulering onderstreept (Hacker, 2023). Hacker (2023) stelt voor dit aan te pakken via een combinatie van co-regulering, duurzaamheid-by-design-principes en integratie van AI-verwerkingsprocessen in het Emissions Trading System (ETS). Voor organisaties zoals Turtle Srl betekent dit dat verduurzaming van LLM-integratie niet alleen een intern optimalisatietraject is, maar ook een manier om voorbereid te zijn op toekomstige rapporterings- en conformiteitseisen (Hacker, 2023). Bovendien kan het expliciet meenemen van duurzaamheid in architectuur- en inkoopbeslissingen bijdragen aan het imago en de geloofwaardigheid van bedrijven die zelf actief zijn rond ecologische en sociale transitie, zoals in deze casus het geval is.

Naast de formele regelgeving groeit ook het aantal vrijwillige transparantie-initiatieven vanuit de sector zelf. Initiatieven zoals de AI Energy Score van Hugging Face beogen een gestandaardiseerde, publiek vergelijkbare maatstaf voor het energieverbruik van taalmodellen te bieden, zodat gebruikers en inkopers modellen kunnen selecteren op basis van hun energieprofiel naast hun prestatiecijfers (Luccioni e.a., 2025). Sapkota e.a. (2025) wijzen erop dat de mate van openheid en transparantie sterk verschilt tussen providers: open-source modellen laten een directe audit van architectuur en verwerkingsstrategie toe, terwijl gesloten modellen als “zwarte dozen” functioneren waarbij de duurzaamheidsrapportering afhankelijk is van de bereidwilligheid van de provider. Voor bedrijven zoals Turtle Srl, die hun eigen duurzaamheidsmissie willen naleven, levert de keuze voor transparante open-source modellen bijgevolg niet alleen een technisch maar ook een governancevoordeel op.

2.8. Synthese en positionering van deze bachelorproef

Samenvattend laat de literatuur zien dat LLM's een substantiële ecologische voetafdruk hebben, dat lokale en cloudgebaseerde inferentie elk specifieke voor- en nadelen kennen en dat factoren zoals elektriciteitsmix, infrastructuur en architecturale keuzes de uiteindelijke impact sterk sturen; aanvullend wordt voor groot-schalige datacenters ook de watervoetafdruk als relevante indicator naar voren geschoven (Li e.a., 2023; Morrison e.a., 2025; Singh e.a., 2024; Wan e.a., 2023). Er bestaan verschillende tools en raamwerken om energieverbruik en CO₂-uitstoot te meten of te schatten, maar de resultaten zijn gevoelig voor de gekozen aannames en de beschikbaarheid van transparante gebruiksgegevens, vooral in cloudscenario's (Bannour e.a., 2021; Fu e.a., 2024; Krupp e.a., 2025; Lannelongue e.a., 2021). Tegelijkertijd komen water- en andere milieu-indicatoren in beeld, wat de analyse complexer maar ook vollediger maakt (Y. Jiang e.a., 2025; Lee e.a., 2024); de afbakening van het kwantitatieve meetluik is reeds toegelicht in de inleiding van dit hoofdstuk.

De literatuur identificeert een aantal duidelijke onderzoekslacunes. Er is nog relatief weinig empirisch werk dat de afwegingen tussen lokale en cloud-LLM-inferentie systematisch onderzoekt binnen concrete webtoepassingen van KMO's, rekening houdend met realistische werklasten, kosten, performantie en de specifieke elektriciteitsmix van verschillende landen (G. Bai e.a., 2024; Fernandez e.a., 2025; Järvenpää e.a., 2023; Wu e.a., 2021). Evenmin is er veel praktijkgericht onderzoek dat technische meetopstellingen koppelt aan een bruikbaar beslissingskader dat ontwikkelaars en technische beslissers helpt om, vertrekkend van hun eigen context, een onderbouwde keuze te maken tussen lokale, cloudgebaseerde of hybride integratie (Singh e.a., 2024; Wan e.a., 2023).

Concreet zijn drie lacunes bijzonder relevant voor deze bachelorproef. Ten eerste ontbreekt het aan empirisch onderzoek dat lokale en cloud-LLM-inferentie vergelijkt voor het specifieke toepassingsscenario van gestructureerde documentextractie, een taak met een onderscheidend belastingsprofiel (lange input, korte gestructureerde output) dat de energiedynamieken anders stuurt dan een chatbot (Fernandez e.a., 2025). Ten tweede is er weinig aandacht voor de impact van de Italiaanse elektriciteitsmix op de ecologische afweging, terwijl dit de lokale opstelling significant benadeelt ten opzichte van landen met een groenere stroommix (Ember, 2026). Ten derde is er geen gepubliceerd beslissingskader dat ontwikkelaars bij KMO's of softwarebedrijven stap voor stap begeleidt bij hun keuze, rekening houdend met privacy, volume, budget en duurzaamheid tegelijkertijd (Singh e.a., 2024; Wan e.a., 2023).

Deze bachelorproef speelt in op die lacunes door in de context van Turtle Srl een proof-of-concept webomgeving op te zetten, waarbij lokale en cloudgebaseerde LLM-integratie experimenteel worden vergeleken in termen van energieverbruik, geschatte CO₂-uitstoot, responstijd, kosten en betrouwbaarheid, met expliciete

aandacht voor de Italiaanse en Belgische context (Ember, 2026; Morrison e.a., 2025; Singh e.a., 2024). De in dit hoofdstuk geschetste inzichten worden in de methodologie vertaald naar concrete keuzes rond experimenteel ontwerp, meetstrategie en analyse, en vormen uiteindelijk de basis voor het beslissingskader dat de doelgroep moet ondersteunen bij duurzame AI-integratie in webontwikkeling.

3

Methodologie

In dit hoofdstuk wordt de gevolgde methodologie toegelicht. Het onderzoek hanteert een vergelijkende experimentele aanpak op basis van de casus bij Turtle Srl. De kern bestaat uit het opzetten van een proof-of-concept webomgeving waarin interne prototypes voor documentextractie en stuklijstanalyse zowel met een lokaal gehost als met een cloudgebaseerd LLM-systeem worden aangestuurd. Het stappenplan is opgedeeld in zes fasen, gespreid over twaalf weken. Elke fase heeft een duidelijk doel, een aanpak en opleveringen.

3.1. Fase 1: Literatuur en casusverkenning (week 1-2)

In de eerste fase van de methodologie wordt de fundering gelegd voor het empirisch onderzoek door de inzichten uit de literatuurstudie te vertalen naar concrete keuzes voor de testomgeving. In overleg met Turtle Srl is besloten om het onderzoek uitsluitend te richten op de *inferentiefase* van Large Language Models voor documentextractie; modeltraining of verfijning valt buiten de scope.

Het experimentele ontwerp maakt gebruik van de intern ontwikkelde applicatie *ExTest*. Dit framework dient als een blauwdruk voor de uiteindelijke testomgeving en is geoptimaliseerd voor het meten van zowel lokale als cloud-gebaseerde inferentie. De resultaten van deze fase zijn de selectie van de meetinstrumenten (Code-Carbon voor lokaal, EcoLogits voor cloud) en de definitieve shortlist van modellen die in de experimenten zullen worden getest. Deze componenteselectie wordt uitgebreid besproken in Hoofdstuk 4.

3.2. Fase 2: Requirementsanalyse en experimenteel ontwerp (week 3-4)

In fase 2 worden de functionele en niet-functionele requirements voor de testomgeving vastgesteld. Dit gebeurt via diepgaande gesprekken met de co-promotor

en technische stakeholders bij Turtle Srl om wensen rondom functionaliteit, kwaliteit, privacy en performantie te verduidelijken.

Op basis van deze behoeften wordt het experimentele ontwerp en het meetkader uitgewerkt. Dit omvat de definitie van de KPI's (zoals het regelgebaseerde 100-puntensysteem voor extractiekwaliteit en het Break-Even Point) en de configuratie van het experimenteel protocol. De gedetailleerde uitwerking van deze requirements en de definitie van de meetwaarden zijn terug te vinden in Hoofdstuk 5. De oplevering van deze fase is het formele experimentele protocol.

3.3. Fase 3: Implementatie van de testomgeving (week 5–6)

In fase 3 wordt de PoC webomgeving opgezet. De nadruk ligt op het reproduceren van de relevante werkstromen binnen de ExTest-applicatie. De omgeving bestaat uit een webfrontend voor het uitvoeren van testscenario's en een backend die zowel het lokale model als de cloud-API's kan aanspreken via een uniforme interface. Logging- en meetcomponenten registreren per verzoek alle relevante gegevens. Belangrijk in deze fase is de integratie van de instrumentatie-bibliotheken: CodeCarbon voor de directe hardwaremetingen bij lokale inferentie (GPU/CPU/RAM) en EcoLogits voor de schatting van de cloud-impact via API-interceptie. De oplevering is een werkende testomgeving waarin dezelfde workflow via beide paden kan worden uitgevoerd, inclusief de registratie van de ecologische voetafdruk per verzoek. Al deze implementatiedetails worden besproken in Hoofdstuk 6.

3.4. Fase 4: Simulatie en dataverzameling (week 7–9)

In fase 4 worden de gesimuleerde werklasten uitgevoerd volgens het vastgelegde protocol. Scenario's worden gebaseerd op een geanonimiseerde testdataset van Italiaanse energiefacturen met variërende lay-out en documentcomplexiteit. Batches van verzoeken worden naar zowel de lokale als de cloudvariant gestuurd, waarbij per verzoek tijdstempels, responstijden, tokentellingen en kenmerken van in- en uitvoer worden gelogd. Het energieverbruik wordt real-time gecapteerd (CodeCarbon voor lokaal, EcoLogits voor cloud) en de extractieresultaten worden na afloop structureel gevalideerd via een regelgebaseerd 100-puntensysteem. Een inhoudelijke vergelijking met de ground truth is voorzien als post-hoc stap buiten de geautomatiseerde testprocedure. De uitvoering en opslag van de resultaten worden in detail besproken in Hoofdstuk 7. De oplevering is een gestructureerde dataset met alle ruwe meetwaarden en accuracy-scores per model en document.

3.5. Fase 5: Analyse en interpretatie (week 10–11)

In fase 5 wordt de verzamelde data geanalyseerd om patronen te identificeren. Er worden gemiddelden en spreidingen berekend voor alle gedefinieerde KPI's. Een belangrijk onderdeel is de break-even-analyse, waarbij de operationele en afgeschreven kosten van de lokale infrastructuur worden afgezet tegen de cloud-tarieven. Tevens wordt de invloed van de regionale stroommix (Italië vs. België) geëvalueerd. De opleveringen zijn representatieve tabellen, grafieken en een eerste set conclusies.

3.6. Fase 6: Beslissingskader en rapportering (week 12)

In de laatste fase worden de analytische bevindingen vertaald naar een praktisch beslissingskader voor IT-architecten. Dit resulteert in een beslissingsboom die technische beslissers ondersteunt bij hun infrastructuurkeuze op basis van volume, privacy en duurzaamheid. De uiteindelijke opleveringen zijn dit beslissingskader en de afgewerkte thesis.

3.7. Beperkingen van het Onderzoek

Dit onderzoek hanteert bewuste scopeafbakening en instrumentele keuzes die de reikwijdte en nauwkeurigheid van de resultaten beïnvloeden. Transparantie over deze beperkingen is essentieel voor een correcte interpretatie van de bevindingen.

Uitsluiting van netwerkennergie en embodied carbon.

De systeemgrenzen van dit onderzoek omvatten uitsluitend de operationele IT-load van de inferentiehardware. De energiekost van datatransmissie via het internet (voor cloud-aanroepen) en de *embodied carbon* van de hardware (de productiemissies van GPU, moederbord en koelinfrastructuur) zijn bewust buiten beschouwing gelaten. Dit is een pragmatische afbakening die haalbaar is binnen de beperkingen van een on-premise PoC met beperkte instrumentatiemiddelen. Methodologisch leidt deze keuze ertoe dat de cloud-emissies licht in het voordeel worden getrokken ten opzichte van de werkelijke totale voetafdruk: het datatransport voor cloud-aanroepen verbruikt immers eveneens energie die hier niet wordt meegeteld (Morrison e.a., 2025; Wu e.a., 2021). Resultaten mogen bijgevolg niet worden geïnterpreteerd als een volledig levenscyclusanalytische vergelijking; de vergelijking dekt uitsluitend het operationele inferentieluik.

Meetafwijking van CodeCarbon.

De lokale energiemetingen zijn gebaseerd op CodeCarbon, dat via software-instrumentatie (NVIDIA Management Library (NVML) voor de GPU en Running Average Power Limit (RAPL) voor de CPU) het hardware-energieverbruik benadert.

Externe validatiestudies tonen aan dat dergelijke software-gebaseerde tools structurele afwijkingen tot circa 40% kunnen vertonen ten opzichte van directe hardwaremetingen met fysieke stroommeters (Fischer, 2025). De absolute waarden van de lokale energiemetingen dienen bijgevolg als schattingen te worden beschouwd, niet als exacte elektrotechnische metingen. De relatieve rangschikking tussen modellen en de richting van het cloud-lokaal verschil zijn robuuster voor systematische fouten, aangezien alle lokale modellen via dezelfde instrumentatiemethode zijn geëvalueerd. Deze beperking is inherent aan het gebruik van softwaregebaseerde instrumentatie in een toegepaste IT-context; fysieke stroommeters op component-niveau waren buiten het bereik van dit onderzoek.

Gebruik van heuristieken.

De lokale PUE-factor van 1,08 is empirisch bepaald via een reeks gecontroleerde metingen op de workstation van Turtle Srl (zie Sectie 4.1.2 voor een uitgebreide uitleg en Sectie 6.3.2 voor de concrete berekening), en benadert de werkelijke waarde voor deze specifieke configuratie. Deze waarde is een doelbewuste methodologische benadering, geen microscopisch precieze elektrotechnische meting, en wordt in de resultaten consequent als zodanig gerapporteerd.

Onzekerheid op de cloud-energieschatting via de prefill-correctiefactor.

De cloud-energiewaarden in dit onderzoek zijn gebaseerd op EcoLogits-schattingen die gecorrigeerd worden met een factor κ (Vergelijking 5.2) om de prefill-fase te verwerken. De correctieparameter $\alpha = 0,175$ is empirisch afgeleid uit de scenariovergelijkingen van Caravaca e.a. (2025): uit hun dataset van 149 modellen levert 9× meer inputtokens gemiddeld 2,19× meer energie op, waaruit $\alpha \approx 0,175$ volgt. Twee inherente beperkingen zijn hierbij van kracht. Ten eerste konden Caravaca e.a. (2025) de prefill- en decode-fase *niet* afzonderlijk meten: de *continuous batching*-mechaniek van vLLM verhindert een strikte fasescheiding per request, waardoor α indirect is bepaald via scenario-ontwerp en niet via directe energiemeting per fase. Ten tweede is de relatie tussen tokenvolume en energieverbruik niet-lineair en modelafhankelijk; de afgeleide α is een gemiddelde over diverse architecturen en GPU-types, en wijkt per individueel model af. De gecorrigeerde cloud-energiewaarden ($E_{\text{gecorrigeerd}}$) moeten bijgevolg worden geïnterpreteerd als een *empirisch gemotiveerde bovengrens*, niet als een exacte meting. De ondergrens (ruwe EcoLogits-schatting, $\alpha = 0$) en de gecorrigeerde bovengrens ($\alpha = 0,175$) vormen samen de gerapporteerde bandbreedte. Conclusies over de verhouding lokaal vs. cloud zijn robuust voor de richting van het verschil, maar de absolute cloud-energiecijfers dragen een structurele onzekerheid die groter is dan de meetafwijking van de lokale CodeCarbon-metingen.

Kwaliteitsimpact van kwantisatie.

De lokale modellen worden in dit onderzoek geëvalueerd in gekwantiseerde vorm (Q4_K_M of equivalent), omdat de beschikbare **VRAM** van de RTX 3070 volledig **FP16**-laden van modellen boven 7–8 B parameters verhindert. De cloud-API's draaien vermoedelijk op hogere numerieke precisie. Hierdoor is de vergelijking lokaal vs. cloud niet louter een infrastructuurvergelijking: gemeten kwaliteitsverschillen kunnen deels het gevolg zijn van kwantisatie in plaats van architecturele of modelgrootte-effecten. Zoals uiteengezet in Sectie 2.2.1, is het kwaliteitsverlies bij 4-bit kwantisatie voor de meeste praktische toepassingen beperkt, maar dit dient als nuancering bij de interpretatie van de accuracy-scores meegenomen te worden.

Bedrijfsspecifieke kostenramingen.

De hardware-investering van €1.400 voor de lokale opstelling en de werklustprojecties die worden gehanteerd in de break-even analyse (zie Sectie 8.7) zijn gebaseerd op interne bedrijfsramingen en volumeprojecties van Turtle Srl. Deze waarden zijn niet afgeleid uit een externe marktstudie en zijn niet generaliseerbaar naar andere bedrijfscontexten zonder aanpassing aan het lokale volume, de hardwarekosten en de elektriciteitsprijzen.

4

Componentenselectie: Modellen en Meetinstrumenten

In dit hoofdstuk worden de concrete keuzes en resultaten van de eerste voorbereidende fase toegelicht. Dit omvat de uitgebreide selectie van de meetinstrumenten, de volledige longlist en shortlist van Large Language Models, en de empirische bepaling van de lokale PUE-factor.

4.1. Componentenselectie: Longlist en Shortlist

Om representatieve vergelijkingen te maken, wordt bepaald welke LLM's in het experiment worden opgenomen. Daarnaast wordt er bepaald welke tools er zullen gebruikt worden om de energie- en CO₂-metingen te doen. De selectie van deze componenten is van groot belang, omdat ze de basis vormen voor de vergelijkende analyse en de validiteit van de conclusies.

4.1.1. Selectie van instrumenten voor energie- en CO₂-metingen

Voordat de modellen geselecteerd worden, is het essentieel om de meetinstrumenten vast te leggen die het energieverbruik en de CO₂-uitstoot van zowel lokale als cloudgebaseerde inferentie zullen kwantificeren. In deze bachelorproef worden twee complementaire open-source Python-bibliotheken ingezet: **CodeCarbon** voor lokale inferentie en **EcoLogits** voor cloudgebaseerde inferentie.

CodeCarbon voor lokale inferentie

CodeCarbon is een open-source Python-bibliotheek die de CO₂-uitstoot van rekenprocessen inschat door het werkelijke elektriciteitsverbruik van hardware te meten (Courty e.a., 2024; Lacoste e.a., 2019). De kernformule is:

$$\text{CO}_2\text{-uitstoot} = \text{energieverbruik (kWh)} \times \text{koolstofintensiteit (kgCO}_2\text{/kWh)} \quad (4.1)$$

CodeCarbon meet het **werkelijke vermogen (in Watt)** van drie hardwarecomponenten via afzonderlijke mechanismen (CodeCarbon Contributors, [g.d.](#)). Voor NVIDIA GPU's wordt de NVML aangesproken via `nvidia-ml-py`, die het werkelijke vermogenverbruik in real-time uitleest via `nvmlDeviceGetPowerUsage()`. Dit werkt met alle NVIDIA-kaarten, inclusief de Ampere-gebaseerde RTX 3070 van Turtle Srl. Voor CPU-verbruik volgt CodeCarbon een hiërarchie: eerst Intel RAPL op Linux (via het bestandssysteem `/sys/class/powercap/intel-rapl/`), dat zowel Intel als AMD-processors ondersteunt; vervolgens Intel Power Gadget op macOS; en als terugval een Thermal Design Power (maximaal warmtevermogen van een processor) (TDP)-gebaseerde schatting aan de hand van een interne database. RAM-verbruik wordt geschat op basis van een heuristiek van circa 5 watt per Dual Inline Memory Module (DIMM)-slot (CodeCarbon Contributors, [g.d.](#)).

De integratie met Python-workflows verloopt via drie patronen: een `@track_emissions`-decorator, een `with EmissionsTracker()`-contextmanager, of een expliciet start/stop-object (Courty e.a., 2024). In dit onderzoek wordt voornamelijk de contextmanager gebruikt, zoals te zien in Codefragment 4.1.

```
1 import requests
2 from codecarbon import EmissionsTracker
3
4 # De tracker registreert het verbruik tijdens de uitvoering van het
   blok
5 with EmissionsTracker() as tracker:
6     # Voer LLM-inferentie uit via de llama-server HTTP API
7     response = requests.post(
8         "http://localhost:8080/v1/chat/completions",
9         json={"model": "llama-3.1-8b", "messages": [{"role": "user",
10             "content": "..."}]}
```

Codefragment 4.1: Illustratie van de integratie van CodeCarbon via de Python contextmanager, met de llama-server HTTP API als inferentie-backend.

Het is belangrijk de beperkingen van CodeCarbon te erkennen. Recente validatiestudies toonden afwijkingen tot circa 40% (Fischer, 2025). Ondanks deze onnauwkeurigheid is er toch voor CodeCarbon gekozen omdat het de enige tool is die directe uitsplitsing per hardwarecomponent toestaat binnen een Python-omgeving, en omdat de relatieve verschillen tussen modellen nog steeds waardevolle inzichten bieden voor de break-even analyse. Bovendien zorgt de wijdverspreide adoptie in de academische wereld voor een betere vergelijkbaarheid met bestaande literatuur.

EcoLogits voor cloudgebaseerde inferentie

EcoLogits gebruikt een **bottom-up levenscyclusanalyse (LCA)-model** om de milieu-impact van cloud-API-aanroepen te *schatten* (Berthelot e.a., 2024; GenAI Impact, g.d.-a). De bibliotheek onderschept API-responses via functie-patching en berekent het energieverbruik via een parametrisch model dat schaaft met het aantal outputtokens en de actieve parameters (GenAI Impact, g.d.-a). EcoLogits rapporteert vier milieucriteria: energieverbruik (kWh), broeikaspotentieel (kgCO₂eq), abiotische uitputting (kgSbeq) en primaire energie (MJ) (Rincé & Banse, 2025b).

Systeemgrenzen en PUE-normalisatie

Een essentieel aspect van de EcoLogits-methodologie is het gebruik van een vaste factor voor de PUE om de energieconsumptie van de ondersteunende infrastructuur (zoals koeling en stroomvoorziening) in het datacenter te verrekenen. EcoLogits hanteert hiervoor een standaardrange van **1.09** tot **1.14** voor HuggingFace Hub (GenAI Impact, g.d.-a). Deze waarde is gebaseerd op een uitgebreide analyse van de infrastructuur van HuggingFace's cloudprovider en vertegenwoordigt een gemiddelde efficiëntie van hun datacenters (Rincé & Banse, 2025a).

Deze waarde fungeert als de referentiewaarde voor de systeemgrens van de cloud-inferentie in dit onderzoek. Om de vergelijking met de lokale inferentie valide te maken, dient de lokale PUE (die de inefficiënties van de on-premise voeding en koeling vertegenwoordigt) op een vergelijkbare wijze te worden geadresseerd (zie Sectie 4.1.2). Het expliciet rapporteren van deze factor is noodzakelijk om aan te tonen dat de cloud-metingen geen "bare-metal"cijfers zijn, maar een verzwaarde schatting die de volledige facilitaire overhead omvat (GenAI Impact, g.d.-a).

4.1.2. Empirische Normalisatie: Berekening van de Lokale PUE

Een kritieke methodologische uitdaging bij het vergelijken van CodeCarbon-data (lokale inferentie) met EcoLogits-data (cloud inferentie) is het verschil in meetbereik. CodeCarbon registreert via software-instrumentatie (NVML en RAPL) uitsluitend het bare-metal elektriciteitsverbruik van de CPU, GPU en het RAM. Het is echter blind voor de zogenaamde *facility overhead*, zoals de inefficiëntie van de computervoeding (Power Supply Unit (computervoeding) (PSU)), het moederbord en de actieve koelingsventilatoren. EcoLogits daarentegen, rekent voor cloud-inferentie standaard PUE in (GenAI Impact, g.d.-a).

Om een zuivere, genormaliseerde vergelijking op systeemniveau mogelijk te maken, wordt in dit onderzoek de lokale PUE van de workstation empirisch vastgesteld. PUE is de internationaal geaccepteerde en gestandaardiseerde metriek om de energie-efficiëntie van IT-infrastructuur uit te drukken, en wordt gedefinieerd als de verhouding tussen de totale energiebehoefte van een faciliteit en de netto energiebehoefte van de IT-apparatuur (Cho e.a., 2024; Lei e.a., 2022). De formule

luidt als volgt:

$$PUE = \frac{\text{Totaal Systeemverbruik}}{\text{IT Componenten Verbruik}} \quad (4.2)$$

Voor de lokale workstation vertaalt zich dit naar de volgende experimentele opstelling:

1. **Totaal Systeemverbruik:** Gemeten via een fysieke energiemeter aan het stopcontact (*wall-plug power*) gedurende een representatieve, zware inferentiebatchtest waarbij de GPU continu wordt belast.
2. **IT Componenten Verbruik:** Gelijktijdig gemeten via de CodeCarbon `EmissionsTracker` gedurende exact dezelfde batchtest.

Door het gemiddelde stopcontactvermogen te delen door het geaggregeerde CodeCarbon-vermogen, wordt de specifieke lokale PUE-multiplier berekend (Cho e.a., 2024). Indien de workstation bijvoorbeeld 350 Watt aan het stopcontact trekt, terwijl CodeCarbon slechts 260 Watt registreert, bedraagt de lokale PUE 1.34.

Deze empirisch vastgestelde multiplier wordt vervolgens structureel toegepast op alle lokale CodeCarbon-resultaten. Dit garandeert dat de uiteindelijke vergelijking voor de *Break-Even Point* analyse een reële representatie is van "Stopcontact vs. Datacenter", in plaats van de methodologisch foutieve "Naakte Chip vs. Datacenter" vergelijking.

Voor de lokale workstation van Turtle Srl is deze empirische PUE vastgesteld op **1.08**, wat betekent dat de werkelijke energieconsumptie van de lokale inferentie 8% hoger is dan wat CodeCarbon direct meet. Deze formule voor deze waarde wordt in detail besproken in 6.3.2. Een PUE van **1.08** waarde wordt beschouwd als een "goede PUE" voor een on-premise workstation (Phillips, 2022). Deze waarde zal consequent worden toegepast op alle lokale metingen om een eerlijke vergelijking met de cloud-inferentie mogelijk te maken.

4.1.3. Selectie van cloud- en lokale LLM's

De lokale inferentie-opstelling bij Turtle Srl is gebaseerd op een workstation uitgerust met een NVIDIA GeForce RTX 3070 GPU. Deze kaart is gebaseerd op de Ampere-architectuur van NVIDIA, die door de introductie van derde generatie Tensor Cores aanzienlijke voordelen biedt voor AI-inferentie (NVIDIA Corporation, 2020). Hoewel het geciteerde whitepaper betrekking heeft op de GA102-chip (RTX 3090- en RTX 3080-serie), deelt de RTX 3070 via zijn GA104-chip dezelfde Ampere-generatie- en architectuurprincipes; een afzonderlijke architectuurpublicatie specifiek voor de GA104 is publiek niet beschikbaar. De technische specificaties van de RTX 3070 (shader-processors, VRAM, TDP) zijn te raadplegen via de GPU-specificatiedatabank van TechPowerUp (TechPowerUp, 2020). Om de geselecteerde modellen effectief te kunnen draaien binnen de beschikbare 8 GB

VRAM, wordt gebruikgemaakt van kwantisatie (zie Sectie 2.2.1 voor een uitgebreide theoretische toelichting).

Component	Specificaties
CPU	Intel Core i7-9800X (3,80 GHz, 8 cores, LGA2066)
RAM	4× 8 GB Samsung M378A1K43DB2-CTD (DDR4-2666, 32 GB totaal)
GPU	NVIDIA GeForce RTX 3070 (8 GB VRAM, GA104, Ampere)

Tabel 4.1: Hardwareconfiguratie van de lokale inferentie-opstelling bij Turtle Srl.

Longlist

De longlist bevat een overzicht van de initieel overwogen modellen en providers. De focus ligt hierbij op modellen in de 7B-9B parameterklasse, aangezien deze optimaal presteren op de beschikbare 8 GB VRAM van de RTX 3070. Hoewel de RTX 3070 krachtiger is dan zijn voorgangers, blijft het videogeheugen de limiterende factor voor de parametergrootte en de contextlengte bij lokale inferentie.

Cloud-LLM's en providers:

- **OpenAI** (GPT-4o, GPT-4o-mini): Krachtige, breed toepasbare modellen.
- **Google Gemini** (Gemini 1.5 Flash, Gemini 1.5 Pro): Direct ondersteund door EcoLogits.
- **NVIDIA Inference Microservices (NIM)**: Biedt geoptimaliseerde inferentie-endpoints voor open-source modellen zoals Llama 3.1 en Nemotron via een OpenAI-compatibele API.
- **Together AI**: Een cloud-provider die toegang biedt tot de nieuwste Qwen-modellen met hoge throughput.
- **Anthropic**: Claude 3.5 Sonnet.
- **HuggingFace**: Een AI-bedrijf en een open-source gemeenschap die een breed scala aan modellen aanbiedt via hun Inference API.

Lokale LLM's (via llama.cpp):

- **Meta Llama 3.1 8B Instruct**: Op een breed spectrum van taken scoort dit model competitief met grotere propriëtaire systemen. Kenmerkend zijn de ingebouwde ondersteuning voor meerdere talen, programmeercode, gestructureerd redeneren en het aansturen van externe hulpmiddelen (Grattafiori e.a., 2024).
- **NVIDIA Nemotron-Nano-9B-v2**: Dit model onderscheidt zich door een gunstige verhouding tussen redeneernauwkeurigheid en verwerkingssnelheid: voor zijn parameterklasse haalt het op redeneer-benchmarks resultaten die

gelijkwaardig of beter zijn dan andere 9B-modellen, bij een relatief hoge doorvoersnelheid (NVIDIA e.a., 2025).

- **Qwen/Qwen3-VL-8B-Instruct:** Dit multimodale model is in staat zowel visuele als tekstuele invoer te verwerken en behaalt brede benchmarkresultaten dankzij een uitgebreid pretrainingskorpus. In dit onderzoek wordt uitsluitend de tekstmodaliteit benut (S. Bai e.a., 2025).
- **Mistral 7B v0.3:** Dit model behaalt een hoge inferentiesnelheid door gecombineerde aandachtsmechanismen die zowel parallelle verwerking als langere invoersequenties faciliteren (A. Q. Jiang e.a., 2023).
- **Apertus 8B Instruct-2509:** Dit model richt zich op twee tekortkomingen van gangbare open-source modellen: de beperkte meertalige dekking en het gebruik van trainingsdata met privacygevoelige inhoud. Het trainingskorpus is samengesteld uit uitsluitend open, gelicentieerde bronnen waaruit persoonsidentificeerbare informatie systematisch is verwijderd (Apertus e.a., 2025).

De RTX 3070 beschikt over 8 GB VRAM, wat betekent dat modellen groter dan ongeveer 9B parameters niet kunnen worden uitgevoerd zonder significant prestatieverlies (Local AI Master Research Team, 2026). Daarom is de longlist beperkt tot modellen in de 7B-9B klasse, die met behulp van kwantisatie nog steeds binnen deze limiet kunnen passen.

Shortlist en Uiteindelijke Selectie

De shortlist wordt bepaald aan de hand van drie harde requirements. Elk model uit de longlist moet aan de volgende eisen voldoen:

- **(R1)** Tot de 7B-9B parameterklasse behoren zodat het past binnen het beschikbare VRAM;
- **(R2)** Beschikbaar zijn via een ondersteunde cloud-provider als overeenkomstig koppel;
- **(R3)** Een publiek beschikbare gekwantiseerde GGUF-versie hebben.

Tabel 4.2 geeft een overzicht van alle longlist-modellen afgezet tegen deze requirements.

Model	R1	R2	R3	Geselecteerd
Llama 3.1 8B Instruct	•	•	•	•
Apertus 8B Instruct-2509	•	•	•	•
Qwen3-VL-8B-Instruct	•	•	•	•
Mistral 7B v0.3	•	-	•	-
Nemotron-Nano-9B-v2	•	-	•	-

Tabel 4.2: Requirements-matrix voor de modelselectie. (• = requirement aanwezig; - = requirement niet aanwezig)

Mistral 7B v0.3 en NVIDIA Nemotron-Nano-9B-v2 voldoen aan R1 en R3, maar zijn niet beschikbaar als overeenkomstig koppel via de ondersteunde cloud-providers ([HuggingFace \(HF\)](#) Inference API, Together AI en NVIDIA [NIM](#)). Zonder een corresponderende cloud-instantie is een infrastructuurvergelijking op basis van hetzelfde model niet mogelijk, waardoor beide modellen uit de selectie vallen.

De uiteindelijke shortlist bestaat uit de drie modellen die alle requirements halen. Exact hetzelfde model wordt zowel via een cloud-provider als lokaal (in een gekwantiseerde versie) getest, waardoor de invloed van modelarchitectuur op de resultaten wordt geminimaliseerd en de impact van de infrastructuurkeuze wordt geïsoleerd.

Model Familie	Cloud Provider (HF)	Lokale Versie (GGUF)
Llama 3.1	meta/llama-3.1-8b-instruct (HF)	Meta-Llama-3.1-8B-Instruct-Q4_K_M
Apertus	swiss-ai/Apertus-8B-Instruct-2509 (HF)	apertus-8b-instruct-2509-q4_k_m
Qwen3	Qwen/Qwen3-VL-8B-Instruct (HF)	Qwen3-VL-8B-Instruct-Q4_K_M

Tabel 4.3: Definitieve modelselectie en vergelijkingsparen.

Onderbouwing van de keuze:

1. **Meta Llama 3.1 8B Instruct:** Dit model dient als de primaire referentiewaarde. Lokaal wordt de **Q4_K_M** kwantisatie gebruikt, die een optimale balans biedt tussen model-precisie en [VRAM](#)-gebruik op de RTX 3070.
2. **Apertus 8B Instruct-2509:** Apertus is een opkomend model dat zich richt op instructie-gebaseerde taken. Door de cloud-versie te vergelijken met de lokale **Q4_K_M** versie, wordt een interessant alternatief getoetst dat mogelijk beter presteert op specifieke extractietaken.
3. **Qwen/Qwen3-VL-8B-Instruct:** De Qwen-serie van Alibaba Cloud staat momenteel bekend als een van de sterkste open-source alternatieven voor vision-

language taken, maar ook voor text-only inferentie. Door de cloud-versie via **HF** te vergelijken met de lokale **Q4_K_M** versie op de RTX 3070, wordt een modern en krachtig alternatief getoetst.

Hoewel de geselecteerde vergelijkingsparen gebaseerd zijn op dezelfde fundamentele modelarchitectuur (bijv. Llama 3.1 8B), is het essentieel te benadrukken dat dit geen perfecte 1-op-1 hardwarevergelijking betreft, maar een vergelijking van *architecturale implementaties*.

Cloud providers serveren deze modellen doorgaans op **FP16** precisie, terwijl de lokale opstelling op de RTX 3070 gebruikmaakt van sterk gekwantiseerde varianten (Q4_K_M of Q5_K_M). Aangezien kwantisatie niet enkel het **VRAM**-gebruik beperkt, maar ook de benodigde **FLOPSs** en het energieverbruik per matrixvermenigvuldiging (**GEMV**) aanzienlijk verlaagt, voert het lokale model wiskundig gezien een aanzienlijk lichtere taak uit. De vergelijking moet daarom strikt geïnterpreteerd worden als: *Standaard Cloud **API** (ongekwantiseerd) versus Geoptimaliseerde Lokale Edge (gekwantiseerd)*.

Om er zeker voor te zorgen dat er zo weinig mogelijk **VRAM** overblijft maar dat het model zijn rekenwerk niet begint te offloaden naar de CPU, worden de gekwantiseerde modellen geconfigureerd met een contextlengte van 32768 tokens, wat net binnen de 8 GB limiet van de RTX 3070 valt. Er wordt ook gekozen om de **Key-Value (KV)**-cache te kwantiseren omdat dit een aanzienlijke hoeveelheid **VRAM** bespaart en vrijhoudt voor de context en de modelparameters. Deze configuratie zorgt ervoor dat de lokale inferentie volledig op de GPU kan draaien zonder performancevermindering door CPU-offloading, terwijl het cloud-model in zijn standaardconfiguratie blijft draaien.

5

Requirementsanalyse en Experimenteel Ontwerp

In dit hoofdstuk worden de functionele en niet-functionele requirements voor de testomgeving vastgesteld en uitgewerkt in een formeel experimenteel protocol. Dit omvat de definitie van de meetwaarden en KPI's, de normalisatie van de systeemgrenzen en de volledige configuratie van het experimenteel protocol.

5.1. Requirementsanalyse en Bedrijfscontext

Uit de analyse met Turtle Srl komt *data privacy* naar voren als een harde niet-functionele requirement. In industriële contexten bevatten documenten vaak bedrijfsgevoelige informatie. Dit mandateert automatisch het onderzoek naar de inzetbaarheid van een volledig lokale of hybride oplossing, waarbij privacygevoelige extracties on-premise blijven en generieke of geanonimiseerde taken naar de cloud worden verplaatst. Maar, voor het uitvoeren van dit onderzoek werd het toegelaten om de volledige workflow in beide scenario's te testen, inclusief het versturen van dezelfde documenten naar de cloud-API. Dit is een noodzakelijke vereenvoudiging om een directe vergelijking van de ecologische voetafdruk mogelijk te maken. De resultaten zullen echter worden geïnterpreteerd binnen de context van deze privacy-overwegingen, waarbij wordt erkend dat in een echte bedrijfsomgeving de cloud-variant mogelijk alleen zou worden ingezet voor geanonimiseerde of minder gevoelige taken. Wat het wel mogelijk maakt om de cloud-variant toch te overwegen in de casus van Turtle Srl is het implementeren van een pre-processing stap die gevoelige informatie (zoals klantnamen, adressen, factuurnummers) automatisch anonimiseert voordat de documenten naar de cloud worden gestuurd. Dit zou de privacy-risico's aanzienlijk verminderen en tegelijkertijd de mogelijkheid bieden om de cloud-variant te evalueren op een representatieve werklast. Het im-

plementeren van deze stap valt buiten de scope van dit onderzoek, maar het is een belangrijke overweging voor toekomstige implementaties in een productieomgeving.

5.2. Systeemgrenzen en Meetbereik

Om een wetenschappelijk valide vergelijking te maken tussen lokale infrastructuur en cloud-gebaseerde diensten, is het essentieel om de systeemgrenzen van de metingen strikt te definiëren. Zonder normalisatie zou een directe vergelijking tussen CodeCarbon en EcoLogits leiden tot een vertekend beeld, aangezien beide tools verschillende abstractieniveaus hanteren (Courty e.a., 2026; Rincé & Banse, 2025b).

5.2.1. Definitie van het lokale meetbereik (On-Premise)

De metingen op de lokale workstation in Italië zijn gebaseerd op directe hardware-instrumentatie.

- **Wat wordt gemeten:** Het actuele stroomverbruik (in kWh) van de GPU (via NVML), de CPU (via RAPL) en een schatting van het RAM-geheugen tijdens de actieve inferentie-context (CodeCarbon Contributors, g.d.).
- **Normalisatie (PUE):** Omdat CodeCarbon enkel de "bare-metal"consumptie van de chips meet, wordt de data wiskundig belast met een lokale PUE-factor van 1.08 (zie Sectie 6.3.2 in Hoofdstuk 6). Dit compenseert voor het energieverlies in de voeding (PSU), de moederbord-chipset en de actieve koeling (fans), waardoor de data vergelijkbaar wordt met de verzwaarde cijfers uit clouddatacenters (Wu e.a., 2021).
- **Wat wordt uitgesloten:** Het energieverbruik van de SSD-opslag (buiten de initiële laadfase), randapparatuur zoals monitors, en de algemene netwerkapparatuur (routers/switches) aan de zijde van de workstation. Het verbruik van deze componenten wordt ingeschat door de PUE-factor.

5.2.2. Definitie van het cloud-meetbereik (API)

De cloud-metingen via HF zijn gebaseerd op de *bottom-up* schattingsmodellen van EcoLogits.

- **Wat wordt gemeten:** Het geschatte energieverbruik dat nodig is voor de verwerking van de specifieke token-volumes op enterprise-grade hardware (zoals NVIDIA H100/A100). Dit model omvat inherent de PUE van het datacenter (Rincé & Banse, 2025b).
- **Kritische beperking — inputtokens worden *niet* meegerekend:** De standaard EcoLogits-schatting (`raw_energy_kwh`) is uitsluitend gebaseerd op het

aantal *outputtokens* (de decoding-fase):

$$E_{\text{raw}} = N_{\text{out}} \times e_{\text{token}} \quad (5.1)$$

De energie die wordt verbruikt om de inputprompt te verwerken (de *prefill*-fase) blijft volledig buiten beschouwing in de standaard bibliotheekimplementatie (Rincé & Banse, 2025a). Bij gebruik in chatbots of samenvattingstaken — waarbij input en output in dezelfde orde van grootte liggen — is deze vereenvoudiging aanvaardbaar. Bij documentextractie, waarbij de factuurtekst na preprocessing typisch **100 tot 200 keer meer inputtokens** omvat dan de gestructureerde JSON-output, leidt deze aanname tot een ernstige onderschatting van het werkelijke energieverbruik. Dit is de meest impactvolle methodologische beperking van EcoLogits voor dit specifieke belastingsprofiel en wordt gecorrigeerd via de methode beschreven in §5.2.3.

- **Wat wordt uitgesloten:** De energie geconsumeerd door de netwerk-hop tussen de workstation in Italië en het clouddatacenter. Ook de embodied carbon van de cloud-hardware wordt niet bekeken. Hoewel deze beide deel uitmaken van de totale ecologische voetafdruk, valt de variabiliteit van het internet-traject buiten de gecontroleerde invloedsfeer van dit onderzoek. Omdat er geen publieke data beschikbaar is over de embodied carbon van elke component in de lokale infrastructuur, wordt deze factor in beide scenario's buiten beschouwing gelaten om een eerlijke vergelijking te waarborgen.

5.2.3. Normalisatie van de Werklast (Prefill vs. Decoding)

Een belangrijk aspect van het meetbereik is het onderscheid tussen de *prefill*-fase en de *decoding*-fase. Bij documentextractie uit PDF's is de input-prompt (de factuurtekst) vaak vele malen groter dan de gestructureerde JSON-output.

Aangezien de *prefill*-fase rekenintensief is (parallele verwerking) en de *decoding*-fase bandbreedte-intensief (sequentiële verwerking), rapporteert dit onderzoek de energie-efficiëntie op basis van **Total Tokens (Input + Output)** (Samsi e.a., 2023). Dit voorkomt dat de cloud-variant onterecht efficiënter lijkt door enkel naar de korte output te kijken, terwijl de zware rekenlast van het inlezen van de PDF onzichtbaar zou blijven.

Correctie voor de niet-gemeten prefill-energie (EcoLogits).

Zoals uiteengezet in §5.2.2, schat EcoLogits de energie op basis van uitsluitend outputtokens. De prefill-fase (verwerking van inputtokens) is afwezig uit deze formule. Voor extractietaken met inputprompts van enkele duizenden tot 25 000 tokens en outputs van slechts 60–100 tokens leidt dit tot een ernstige onderschatting.

Om een eerlijke vergelijking met de lokale meting (die beide fasen omvat) mogelijk te maken, wordt een correctiefactor κ toegepast. De grondslag is het ex-

perimentele werk van Caravaca e.a. (2025), dat de totale GPU-energie per prompt meet over een raster van input- en outputlengtes. Uit hun scenariovergelijking over 149 modellen blijkt dat 9× meer inputtokens (bij gelijke outputlengte) gemiddeld slechts 2,19× meer energie oplevert, terwijl 9× meer outputtokens 11× meer energie geeft. Hieruit volgt dat een inputtoken gemiddeld ruwweg 5 à 6× minder energie-intensief is per token dan een outputtoken. Formeel wordt de verhouding uitgedrukt via een parameter α , die aangeeft welk deel van de energie per outputtoken een inputtoken kost. Het energiemodel stelt dat de totale energie evenredig is met $E = \alpha \times N_{\text{in}} + N_{\text{out}}$, waarbij een outputtoken genormaliseerd op 1 staat. De twee scenario's uit Caravaca e.a. (2025) zijn: scenario 1 met 100 input- en 100 outputtokens ($P_1 = 100\alpha + 100$) en scenario 2 met 900 input- en 100 outputtokens ($P_2 = 900\alpha + 100$). De gemeten energieverhouding $P_2/P_1 = 2,19$ levert de vergelijking:

$$\frac{900\alpha + 100}{100\alpha + 100} = 2,19$$

Uitwerken geeft:

$$900\alpha + 100 = 2,19 \times (100\alpha + 100)$$

$$900\alpha + 100 = 219\alpha + 219$$

↓

$$(900 - 219)\alpha = 219 - 100$$

↓

$$\alpha = \frac{119}{681} \approx 0,175$$

Er zijn twee belangrijke kanttekeningen die gelden bij deze afleiding. Ten eerste veronderstelt het lineaire model een vaste α over alle modellen, terwijl Caravaca e.a. (2025) aantonen dat de energie-tokensverhouding niet-lineair verloopt en per model en GPU-type varieert (Figuur 1 in hun paper). Ten tweede konden zij prefill- en decode-energie niet afzonderlijk meten vanwege de *continuous batching*-mechaniek van vLLM, waarbij een request in de prefill-fase kan zitten terwijl een ander in de decode-fase zit; α is bijgevolg indirect afgeleid via scenario-design en niet via directe fasemeting. $\alpha = 0,175$ is een empirische benadering op basis van gemiddelden over 149 modellen, niet een exact vastgestelde constante. De onzekerheid op de gecorrigeerde cloud-energiewaarden is bijgevolg groter dan die op de lokale CodeCarbon-metingen; zie ook Sectie 3.7 voor een expliciete bespreking van deze beperking.

De gecorrigeerde energie per inferentie-aanroep wordt als volgt berekend:

$$E_{\text{gecorrigeerd}} = E_{\text{raw}} \times \kappa, \quad \kappa = \frac{N_{\text{out}} + \alpha \times N_{\text{in}}}{N_{\text{out}}} \quad (5.2)$$

waarbij N_{in} het aantal inputtokens is, N_{out} het aantal outputtokens, en $\alpha = 0,175$. De factor κ schaaft de ruwe schatting omhoog in verhouding tot het inputtokengewicht. Bij een verhouding van 150:1 inputtokens tot outputtokens — representatief voor de langere documenten in deze dataset — geeft dit een correctiefactor van $\kappa = (1 + 0,175 \times 150)/1 = 27,25$.

De empirische verdeling van κ over alle cloud-aanroepen in dit experiment, en de concrete impact ervan op de gerapporteerde cloud-energiewaarden, worden besproken in Sectie 8.2 van Hoofdstuk 8.

5.2.4. Kwantisatie en Functionele Pariteit

Tot slot erkent dit onderzoek de asymmetrie in model-precisie als onderdeel van de systeemgrens. De cloud-modellen draaien op volledige precisie (FP16), terwijl de lokale modellen gebruikmaken van 4-bit of 5-bit kwantisatie. Deze keuze is bewust: het vertegenwoordigt de *praktische realiteit* van een bedrijf dat moet kiezen tussen een kant-en-klare API of een geoptimaliseerde lokale implementatie op beperkte hardware. De winst in energie-efficiëntie door kwantisatie wordt beschouwd als een intrinsiek voordeel van de lokale architectuur (Khan e.a., 2025).

5.2.5. Definitie van Metrics en KPI's

Om een zuivere en methodologisch correcte vergelijking te garanderen, worden prestatie-eisen strikt gescheiden van hardware-efficiëntie. De volgende indicatoren worden gedefinieerd.

Kwaliteit (Structurele Validatie)

Gemeten via een regelgebaseerd 100-puntensysteem dat de geëxtraheerde JSON structureel valideert ten opzichte van het verwachte Pydantic-schema. De score is opgebouwd uit vier lagen:

- aanwezigheid van het `periodes`-array met minstens één element (25 pt),
- exact één facturatieperiode in die array (15 pt),
- aanwezigheid van elk van de vier verplichte velden (`supplier`, `start_date`, `end_date`, `kwh_quantity`): 10 pt per veld, 40 pt in totaal,
- correct datatype per veld (5 pt per veld, 20 pt in totaal).

De score wordt per inferentie-aanroep berekend en opgeslagen in de SQLite-database. Het Pydantic-schema dwingt structuurconformiteit bovendien *a priori* af: lokaal via *grammaticagestuurde decoding* in llama-server, en bij de cloud-API via de `response_format`-parameter.

Inhoudsnauwkeurigheid (Ground-Truth Score)

Gemeten via een veldgewijze vergelijking van de geëxtraheerde waarden ten opzichte van de handmatig geverifieerde correcte waarden in:

ground_truth_expected_fields_full.json. De score is opgebouwd uit vier componenten van 25 punten elk:

- `supplier`: exacte overeenkomst na normalisering (strip/lower), waarbij `null = null` als correct wordt beschouwd,
- `start_date`: overeenkomst na normalisering van datumformaten naar ISO 8601 (ondersteunde invoerformaten: YYYY-MM-DD, DD/MM/YYYY, DD.MM.YYYY),
- `end_date`: idem,
- `kwh_quantity`: numerieke overeenkomst binnen een tolerantie van 0,5% relatief of 0,01 kWh absoluut.

De structurele validatiescore wordt per inferentie-aanroep inline berekend en opgeslagen. De inhoudsnauwkeurigheidsscore wordt retroactief berekend na afloop van de test: alle opgeslagen extractieresultaten worden vergeleken met de handmatig geverifieerde referentiewaarden, waarna de score per rij in de SQLite-databank wordt weggeschreven. Het eindresultaat is identiek: beide scores zijn beschikbaar per inferentierij voor de verdere analyse.

De scoringsfunctie van de inhoudsnauwkeurigheidsscore is formeel gedefinieerd in Vergelijking 5.3. Hierbij zijn $\hat{s}$ en s^* respectievelijk de geëxtraheerde en verwachte leveranciersnaam (na strip/lower-normalisering), $\tilde{d}$ een datum genormaliseerd naar ISO 8601, en $\hat{q}$, q^* de geëxtraheerde en verwachte kWh-waarde. De functie kent per veld 25 punten toe bij een overeenkomst met de referentiewaarde, en 0 punten bij een afwijking; een perfecte extractie levert 100 punten op.

$$S_{GT} = 25 \times \mathbf{1}[\hat{s} = s^*] + 25 \times \mathbf{1}[\tilde{d}_{start} = \tilde{d}_{start}^*] + 25 \times \mathbf{1}[\tilde{d}_{end} = \tilde{d}_{end}^*] + 25 \times \mathbf{1}[|\hat{q} - q^*| \leq \max(0,01, 0,005 |q^*|)] \quad (5.3)$$

Structurele validatie versus inhoudsnauwkeurigheid

Het verschil tussen beide scores is fundamenteel: de structurele score beoordeelt uitsluitend of de geëxtraheerde JSON het verwachte schema respecteert (*is de vorm correct?*), terwijl de inhoudsnauwkeurigheidsscore meet of de feitelijke geëxtraheerde waarden overeenkomen met de werkelijkheid in het document (*is de inhoud correct?*).

UX/Business Gate (Latentie)

Responstijd gemeten in **Time-To-First-Token** (tijd tot het eerste gegenereerde token) (TTFT). Deze metric wordt uitsluitend gebruikt als een binaire drempelwaarde om te bepalen of de architectuur voldoet aan de zakelijke eisen van de gebruiker. TTFT wordt *niet* gebruikt als maatstaf voor hardware-efficiëntie, aangezien de cloud-TTFT zwaar vervuild wordt door netwerklatentie en load-balancer wachtrijen.

Hardware-efficiëntie

Gemeten in doorvoer en energie-intensiteit. Doorvoer wordt berekend als het totale aantal tokens (input en output) gedeeld door de totale inferentietijd (Vergelijking 5.4). Energie-intensiteit wordt uitgedrukt in milliwattuur per 1.000 tokens (Vergelijking 5.5). Het combineren van input- en outputtokens is essentieel om de zware *prefill*-energiepiek bij lange documenten accuraat te verrekenen. Beide afgeleide metrics worden achteraf berekend uit de ruwe kolommen die per inferentie-aanroep worden opgeslagen.

$$\text{tokens/sec} = \frac{\text{total_tokens}}{\text{total_inference_seconds}} \quad (5.4)$$

$$\text{mWh per 1.000 total tokens} = \frac{\text{energy_kwh} \times 10^9}{\text{total_tokens}} \quad (5.5)$$

Kosten en Break-Even Punt

De vergelijking omvat drie kostencategorieën:

- *API-kosten (cloud)*: Berekend op basis van de gepubliceerde tarieven per model (Hugging Face, 2026), met afzonderlijke prijzen voor input- en outputtokens. Alle bedragen zijn oorspronkelijk gepubliceerd in Amerikaanse dollars (USD) en omgerekend naar euro's (EUR) met behulp van de wisselkoers van de Europese Centrale Bank op 8 mei 2026 (EUR 1 = USD 1,1761) (European Central Bank, 2026).

Model	Input (USD)	Output (USD)	Input (EUR)	Output (EUR)
LLaMA 3.1	\$0,0200	\$0,0500	€0,01700	€0,04251
Qwen	\$0,0800	\$0,5000	€0,06802	€0,42513
Apertus	\$0,1000	\$0,2000	€0,08503	€0,17005

Tabel 5.1: API-tarieven per model in USD en EUR (per 1M tokens)

De kost per inferentie-aanroep wordt berekend via Vergelijking 5.6, waarbij p_{in} en p_{out} de respectievelijke input- en outputtarieven per token zijn.

- *Operationele elektriciteitskosten (lokaal)*: Berekend via Vergelijking 5.7: het door CodeCarbon gemeten energieverbruik wordt genormaliseerd met de lokale PUE-factor en vermenigvuldigd met het actuele elektriciteitstarief. Voor p_{kWh} wordt een prijs van €0,2377/kWh gehanteerd, gebaseerd op de gemiddelde elektriciteitsprijs voor niet-huishoudelijke gebruikers in Italië in 2025-S2 (categorie IB (20MWh tot 499 MWh per jaar), prijs exclusief btw en andere terugvorderbare belastingen en heffingen) (Eurostat, 2022).

- *Hardware-afschrijving (lokaal)*: Berekend via lineaire afschrijving (Vergelijking 5.8) op basis van de geraamde totale aankoopprijs van €1.400 voor de hardwarecomponenten vermeld in Tabel 4.1¹. De levensduur wordt geraamd op **14.600 uur**, overeenkomend met 5 jaar bij 8 uur per dag gedurende alle kalenderdagen (8 u/dag × 365 dagen/jaar × 5 jaar). Naast de afschrijving wordt in beide scenario's een idle-elektriciteitskost aangerekend (Vergelijking 5.9) voor de uren dat de machine aanstaat maar geen inferentie uitvoert, waarbij het systeem gemiddeld 60 W verbruikt in rust. In het *standaard* scenario staat de machine 8 uur per dag aan: $60 \text{ W} \times 8 \text{ u/dag} \times 365 \times 5 \text{ jaar} \times \text{€}0,2377/\text{kWh} \approx \text{€}208$. In de *altijd-actieve* variant staat de machine 24 uur per dag aan: $60 \text{ W} \times 24 \text{ u/dag} \times 365 \times 5 \text{ jaar} \times \text{€}0,2377/\text{kWh} \approx \text{€}625$. De afschrijvingsperiode blijft in beide gevallen 14.600 uur. Beide varianten worden besproken in Sectie 8.7.

$$C_{\text{API}} = N_{\text{in}} \times p_{\text{in}} + N_{\text{out}} \times p_{\text{out}} \quad (5.6)$$

$$C_{\text{elektriciteit}} = E_{\text{kWh}} \times \text{PUE} \times p_{\text{kWh}} \quad (5.7)$$

$$C_{\text{afschrijving}} = \frac{\text{€}1.400}{\text{levensduur (uren)}} \times \frac{\text{total_inference_seconds}}{3600} \quad (5.8)$$

$$C_{\text{idle}} = P_{\text{idle}} \times u_{\text{dag}} \times 365 \times n_{\text{jaar}} \times p_{\text{kWh}} \quad (5.9)$$

Om idle-energieverbruik en meetvervuiling te vermijden, worden modellen na elke inferentie-aanroep automatisch uit het VRAM verwijderd. Dit wordt gerealiseerd via de LLAMA_RELOAD_MODEL_PER_CALL-strategie in de LocalServerManager, waardoor elke meting betrekking heeft op één geïsoleerde inferentie zonder stand-by overhead.

Duurzaamheid (Systeemgrenzen)

Het gemeten energieverbruik (CodeCarbon) wordt wiskundig genormaliseerd met een lokale PUE-vermenigvuldiger om te compenseren voor moederbord- en voeding-inefficiënties. Dit garandeert een eerlijke “systeem-tot-systeem”-vergelijking tegenover de EcoLogits-data, die standaard al datacenter-PUE meerekent.

5.2.6. Experimenteel Protocol en Tool-Configuratie

Op basis van de hierboven gedefinieerde metrics en systeemgrenzen wordt het experiment uitgevoerd volgens een gestandaardiseerd protocol. Dit protocol legt

¹Bij gebrek aan de originele aankoopfactuur werd de totale hardwarekost geraamd op basis van een conservatieve schatting verstrekt door Turtle Srl.

de werklastsamenstelling, het aantal iteraties en de configuratie van de meethulpmiddelen vast, zodat alle inferentie-aanroepen (zowel lokaal als via de cloud-API) onder reproduceerbare en vergelijkbare omstandigheden plaatsvinden.

1. **Belastingsprofielen:** Er worden 21 bestanden van verschillende lengtes en complexiteiten gebruikt, variërend van korte facturen van 4 pagina's tot langere documenten van 45 pagina's. Elk bestand wordt per model 50 keer geïnfereerd, wat resulteert in 1.050 inferenties per model en een totaal van 6.300 inferenties over alle 6 modellen (3 in de cloud en 3 lokaal).
2. **Guided JSON en Test-iteraties:** De LLM's worden getest op hun vermogen om valide JSON-output te genereren voor complexe schema's, zowel met als zonder expliciete schema-afdwinging.
3. **Overhead van grammaticagestuurde decodering:** Om de outputlengte perfect gelijk te trekken tussen cloud en lokaal (en zo de energie per token correct te normaliseren), wordt 100% JSON-compliance afgedwongen. Voor de lokale variant gebeurt dit via llama.cpp's GGML Backus-Naur Form (grammatica-formaat voor constrained decoding in llama.cpp) (GBNF) grammatica-engine (ggml-org, 2023), en voor de cloud via de respectievelijke *gestuurde decodering*-backends van de providers. Dit introduceert een meetbare computationele overhead die als "cost of structure" inherent vervat zit in de gemeten energie-intensiteit.

6

Implementatie van de Testomgeving

In dit hoofdstuk wordt de technische realisatie van de proof-of-concept (PoC) testomgeving in detail beschreven. Voor deze implementatie diende de initiële interne applicatie van Turtle Srl, genaamd *ExTest*, uitsluitend als architecturale blauwdruk. Om een eerlijke en wetenschappelijk robuuste vergelijking tussen lokale en cloudgebaseerde taalmodellen mogelijk te maken, werd vrijwel de volledige verwerkingsketen vanaf de basis opnieuw gemodelleerd. De volledige broncode van het herwerkte framework is beschikbaar als open-source repository.¹ Waar de oorspronkelijke ExTest-architectuur voornamelijk leunde op gesloten, premium cloud-modellen die **Optical Character Recognition (OCR)** op gescande documenten uitvoerden of rauwe documenten direct als input konden verwerken (image-to-text en native PDF-verwerking), was een dergelijke aanpak ongeschikt voor de lokale modellen met beperkt geheugen uit onze shortlist. Dit hoofdstuk focust daarom op de vernieuwde, reproduceerbare software-architectuur. De verwerkingsketen omvat het inlezen en voorverwerken van documenten naar een voor kleine modellen hapklaar formaat, de semantische sturing via datamodellen, en de geoptimaliseerde orkestratie van inferentie-aanroepen. Tabel 6.1 geeft een overzicht van de Python-bibliotheken die de kern van het PoC-systeem vormen.

6.1. Documentextractie en Voorverwerking

Zoals eerder aangehaald, stuurde de initiële ExTest-blueprint ruwe documenten of afbeeldingen rechtstreeks naar multimodale premium LLM's (zoals GPT-4o) die zelf instonden voor de visuele tekstextractie. Voor deze PoC, waarbij we lokale model-

¹<https://github.com/poenaru-alexandru/extest-bachelorproef>

Parameter	Waarde
Documentextractie (PDF → Markdown)	<code>pymupdf4llm</code>
Datavalidatie en extractieschema	<code>pydantic ≥ 2.0</code>
Energie-instrumentatie (lokale inferentie)	<code>codecarbon</code>
Energie-instrumentatie (cloud-inferentie)	<code>ecologits</code>
Cloud-inferentie API-client	<code>huggingface_hub</code>
Lokale inferentie HTTP-client	<code>requests ≥ 2.31</code>
Lokale GGUF-runtime (extern binair)	<code>llama-server (llama.cpp)</code>
Omgevingsvariabelen	<code>python-dotenv ≥ 1.0</code>
Resultatenopslag	SQLite via <code>results_db.py</code>

Tabel 6.1: Gebruikte Python-bibliotheken in het PoC-framework

len in de 7B–9B parameterklasse willen evalueren, is dit onmogelijk doordat deze modellen primair tekstueel gedreven zijn en over een aanzienlijk kleiner contextvenster beschikken.

Traditionele tekstextractie via bibliotheken zoals PyPDF2 was geen waardig alternatief: dit resulteert vaak in platte, ongestructureerde datablokken waarbij de visuele en positionele context van tabellen (cruciaal bij facturatiegegevens) verloren gaat, wat slecht is voor de accuraatheid.

Om dit probleem aan te pakken, is binnen het herwerkte framework gekozen voor de integratie van de `pymupdf4llm`-bibliotheek (Artifex Software, Inc., 2024). Deze gespecialiseerde tool fungeert als een deterministische brug: het converteert PDF-documenten direct naar een gestructureerd Markdown-formaat voordat de data naar de LLM wordt gestuurd. Dit proces behoudt de structuur van tabellen, lijsten en sectiekoppen, wat resulteert in een aanzienlijk kwalitatievere *context payload* voor tekst-gebaseerde modellen en het extractieproces fundamenteel vereenvoudigt.

Het is van groot methodologisch belang te benadrukken dat het energieverbruik dat gepaard gaat met deze `pymupdf4llm` documentextractie en voorverwerking *niet* is opgenomen in de gemeten energie-intensiteit van de LLM's. De rationale hiervoor is dat documentextractie een noodzakelijke, deterministische voorwaarde is die inherent verbonden is aan de specifieke applicatie (Document Information Extraction). Deze facilitaire overhead vindt altijd lokaal plaats en vormt het voorbereidende werk waarbinnen de modellen opereren. Door de preprocessing-energie strikt te scheiden van de inferentiemetingen, garandeert het meetsysteem een zuivere en vergelijkbare evaluatie van het *effectieve* energieverbruik van de taalmodellen zelf.

Om redundante parsing te vermijden bij testronden waarbij meerdere modellen

hetzelfde document verwerken, implementeert de `BenchmarkRunner` een in-memory *text cache*. De geëxtraheerde Markdown-tekst wordt gekeyed op het tuple (`bestandspad`, `extractor_naam`) en hergebruikt over alle modelconfiguraties die hetzelfde document verwerken. Dit elimineert onnodige I/O- en CPU-overhead tijdens de meetperiode.

6.2. Datamodellering en Semantische Sturing via Pydantic

Een fundamentele uitdaging bij de inzet van generatieve LLM's voor data-extractie is het afdwingen van een consistente, gestructureerde en parseerbare uitvoer (zoals JSON). Zonder strikte kaders hebben modellen de neiging om overvloedige tekstuele uitleg te genereren of af te wijken van de vereiste datastructuur, wat geautomatiseerde verwerking onmogelijk maakt. Binnen het ExTest-framework wordt `Pydantic` geïmplementeerd om het verwachte extractieschema rigide te definiëren (Pydantic Developers, 2025).

6.2.1. Schemaontwerp en Extractieregels

Het extractiedoelwit wordt gedefinieerd via twee Pydantic-klassen in `Test/modello.py`: het `Periode`-datamodel voor één factuurperiode, en het omvattende `FactuurModel` dat de volledige extractietaak omschrijft. De veldnamen zijn in het Engels opgesteld omwille van brede modelcompatibiliteit. De semantische sturing naar de Italiaanse broncontext wordt gerealiseerd via expliciete extractieregels in de klasse-docstring van `FactuurModel`.

Codefragment 6.1 toont de volledige modeldefinitie. De docstring vervult een dubbele rol: enerzijds genereert Pydantic hieruit het JSON-schema voor de grammaticagestuurde decoding van de LLM, anderzijds fungeert de tekstuele inhoud als een gedetailleerde systeem-instructie. De **CRITICAL RULES** sectie voorkomt dat het model historische verbruikstabellen (zoals *Andamento storico dei prelievi*) verwart met de gefactureerde periode, een veelvoorkomende fout bij kleinere modellen. De **WHERE TO FIND** sectie biedt inhoudelijke richtpunten in het Italiaans (*periodo di fornitura, fornitura dal ...al ...*), waardoor de taalkloof tussen het Engelstalige schema en de Italiaanse brondocumenten verkleint.

6.2.2. Grammaticagestuurde decoding

Het JSON-schema dat Pydantic uit `FactuurModel` genereert via `model_json_schema()`, wordt rechtstreeks meegegeven aan de inferentie-aanroep als `response\format`. Voor lokale modellen via `llama-server` maakt dit gebruik van de ingebouwde grammaticagestuurde decoding van `GGUF`, wat garandeert dat het model structureel valide JSON produceert dat overeenkomt met het schema, ook wanneer het model de structuur niet spontaan zou respecteren (Gerganov & contributors, 2023).

6.3. Implementatie van de Energie- en Emissiemetingen

Het hart van de wetenschappelijke setup is de integratie van meetinstrumentatie om het stroomverbruik en de CO₂eq-uitstoot te kwantificeren. Het framework hanteert twee complementaire meetmechanismen, één voor lokale en één voor cloudgebaseerde inferentie.

6.3.1. Directe Hardware-instrumentatie met CodeCarbon

Voor de lokaal gehoste modellen wordt de `EmissionsTracker` van de CodeCarbon-bibliotheek gebruikt. De tracker wordt gestart onmiddellijk vóór de HTTP-inferentieoproep naar `llama-server` en gestopt zodra het antwoord volledig ontvangen is (zie Codefragment 6.2). CodeCarbon leest het energieverbruik via drie kanalen: `NVML` voor de `GPU` (systeembreed, waardoor ook het verbruik van het externe `llama-server`-proces correct wordt meegeteld), `RAPL` voor de `CPU`, en een `DIMM`-gebaseerde heuristiek voor het `RAM`. Op Linux bepaalt CodeCarbon het `RAM`-verbruik automatisch via hardware-detectie (CodeCarbon Contributors, g.d.); een handmatige overschrijving via `force_ram_power` is hierdoor niet vereist. Het samplinginterval wordt ingesteld op 500 ms (`measure_power_secs=0.5`), wat voldoende detailniveau biedt voor de inferentie-oproepen in dit experiment. Tabel 6.2 toont de drie lokale modellen die in de prestatietest worden geëvalueerd. Alle modellen zijn opgeslagen als gekwantiseerde `GGUF`-bestanden.

Parameter	Waarde
Meta-Llama 3.1 8B Instruct (8,03B)	Meta-Llama-3.1-8B-Instruct-Q4_K_M.gguf
Qwen3-VL-8B-Instruct (9B)	Qwen3VL-8B-Instruct-Q4_K_M.gguf
Apertus-8B-Instruct-2509 (8B)	Apertus-8B-Instruct-2509-Q4_K_M.gguf
Kwantisatiemethode (alle modellen)	Q4_K_M (4-bit mixed)

Tabel 6.2: Geëvalueerde lokale modellen (`GGUF`-formaat)

Tabel 6.3 toont de configuratieparameters die via omgevingsvariabelen aan `llama-server` worden meegegeven.

Tabel 6.4 toont de configuratieparameters van `EmissionsTracker` en de bijbehorende inferentieparameters voor lokale aanroepen.

6.3.2. PUE-correctie en Regionale Emissiefactoren

Het door CodeCarbon gerapporteerde basisverbruik betreft de *IT-load*: de energie die rechtstreeks door de hardware wordt verbruikt (`d.energy_consumed`). Om de overhead van voeding (`PSU`), moederbord-chipset en actieve koeling te ver-

Tabel 6.3: llama-server configuratieparameters (lokale inferentie)

Parameter	Waarde
-ctx-size (contextvenster)	32 768 tokens
-batch-size	2 048
-ubatch-size	2 048
-flash-attn	aan
-n-gpu-layers	-1 (alle lagen op GPU)
-cache-type-k	q4_0
-cache-type-v	q4_0
Poort	8080
Modelherlaadstrategie	per-aanroep isolatie (LLAMA_RELOAD_MODEL_PER_CALL=true)

rekenen, wordt de PUE-factor *niet* via de ingebouwde pue-parameter aan de tracker meegegeven, maar manueel toegepast ná het stoppen van de tracker (zie Codefragment 6.2). Door de PUE-correctie buiten de tracker te houden, blijven `raw_energy_kwh` (de bare IT-load) en `energy_kwh_with_pue` (gecorrigeerd voor facility-overhead) als afzonderlijke waarden beschikbaar voor analyse. Voor de lokale testopstelling wordt een PUE van 1,08 gehanteerd (zie Sectie 6.3.2), wat de praktische overhead van consumentenhardware buiten een datacenteromgeving weerspiegelt.

Berekening van PUE

De empirische PUE van 1,08 is vastgesteld via de methode beschreven in Sectie 4.1.2: het gemeten stopcontactvermogen werd gedeeld door de gelijktijdige CodeCarbon-registratie tijdens een representatieve inferentie-batchtest. Concreet werd de volgende formule toegepast op de waarde gemeten door een slimme energiemeter die op het stopcontact is aangesloten, en de `energy_consumed` gerapporteerd door CodeCarbon voor een batch van 63 inferentie-aanroepen:

$$PUE = \frac{E_{\text{Totaal}}}{E_{\text{IT}}} = \frac{\approx 0,068 \text{ kWh}}{\approx 0,063 \text{ kWh}} \approx 1,08 \quad (6.1)$$

Op basis van dit gecorrigeerde energieverbruik berekent het framework de CO₂eq-uitstoot per regio via hardgecodeerde emissiefactoren. Tabel 6.5 geeft een overzicht van de gebruikte waarden uit Ember (2026).

De rationale voor het opnemen van meerdere regio's is methodologisch: omdat cloudproviders hun datacenters strategisch over de wereld verspreiden, is het essentieel om een spectrum van emissiefactoren te hanteren die verschillende energiemixen weerspiegelen. Dit maakt het mogelijk om de gevoeligheid van de re-

Tabel 6.4: CodeCarbon- en inferentieparameters (lokale inferentie)

Parameter	Waarde
measure_power_secs	0,5s
save_to_file	False
log_level	warning
Energiebronnen	NVML (GPU), RAPL (CPU), DIMM-heuristiek (RAM)
PUE (manueel na meting)	1,08
Inferentie temperature	0
Inferentie max_tokens	1024
Inferentie stream	False
response_format	json_schema (grammaticagestuurde decodering)

Regio	Code	Factor (kgCO ₂ eq/kWh)
Italië	ITA	0.28478
België	BEL	0.14982
Frankrijk	FRA	0.04144
Duitsland	DEU	0.32965
VS	USA	0.3844
Wereld	WOR	0.45829

Tabel 6.5: Regionale emissiefactoren (kgCO₂eq/kWh) gebruikt in het PoC. Gegevens uit Ember (2026)

sultaten voor regionale energieprofielen te analyseren en biedt een meer genuanceerd inzicht in de milieu-impact van cloud-inferentie in verschillende geografische contexten. Zo kan er ook geanalyseerd worden hoe de locatie van de lokale workstation (Italië) zich verhoudt tot de cloud-inferentie in termen van CO₂eq-uitstoot, afhankelijk van de regionale energiemix van de cloudprovider.

6.3.3. Cloud-estimatie via EcoLogits

Voor de cloudgebaseerde inferenties wordt `EcoLogits` ingezet. Van cruciaal belang is de initialisatievolgorde: `EcoLogits.init()` moet worden aangeroepen vóór enige import van `huggingface_hub`, aangezien `EcoLogits` via *functievervanging* de originele klasse moet onderscheppen vóór ze elders in gebruik wordt genomen. Binnen `benchmark_runner.py` gebeurt dit daarom op moduleniveau, vóór alle andere imports:

```
1 EcoLogits.init(providers=["huggingface_hub"])
```

EcoLogits injecteert een `impacts`-object in elke respons van `huggingface_hub`. EcoLogits biedt een volledige LCA-benadering: het `impacts.energy`-veld omvat zowel het directe inferentieverbruik (inclusief datacenter-PUE) als de afgeschreven *embodied carbon* van de cloud-hardware. Om de vergelijkbaarheid met de CodeCarbon-metingen voor lokale runs te waarborgen, wordt uitsluitend `impacts.usage.energy` gebruikt: de *usage*-fase inclusief PUE, maar exclusief embodied carbon. Dit resulteert in een symmetrische vergelijkingsbasis: beide sporen meten de operationele energie gecorrigeerd voor facility-overhead, zonder geamortiseerde hardware-productie-emissies (GenAI Impact, [g.d.-b](#)).

Registratie van aangepaste modellen

EcoLogits beschikt over een ingebouwde modeldatabank met energieparameters voor bekende modellen. Voor modellen die niet in deze databank voorkomen, registreert het framework ze manueel vóór de eerste inferentie (zie Codefragment 6.3). De `parameters`-waarde correspondeert met het aantal miljarden parameters van het model en wordt gebruikt door EcoLogits om het energieverbruik te schatten.

Op basis van de gecorrigeerde energiewaarde berekent `HuggingFaceProvider` vervolgens dezelfde set regionale CO_2eq -projecties als de lokale provider (zie Tabel 6.5), zodat beide meetsporen vergelijkbaar blijven.

Correctie voor de prefill-fase

De methodologische grondslag van deze correctie (de beperking van EcoLogits tot uitsluitend outputtokens en de herleiding van $\alpha = 0,175$ uit de experimentele data van Caravaca e.a. (2025)) wordt uitgewerkt in Sectie 5.2.3 (Vergelijking 5.2). Hieronder wordt beschreven hoe die correctie concreet in de implementatie wordt toegepast.

De ruwe EcoLogits-schatting ($E_{\text{raw}} = N_{\text{out}} \times e_{\text{token}}$) wordt per aanroep gecorrigeerd met de factor κ uit Vergelijking 5.2. Voor een representatieve oproep met $N_{\text{in}} = 9.327$ en $N_{\text{out}} = 66$ geeft dit $\kappa \approx 25,7$. In de resultaten worden beide waarden opgeslagen: `raw_energy_kwh` (onbewerkte EcoLogits-schatting, enkel outputtokens) en `energy_kwh_with_pue` (gecorrigeerde bovengrens, gebruikt voor alle lokaal-cloud-vergelijkingen). De correctiefactor κ zelf wordt opgeslagen als `ecologits_prefill_correction`.

Tabel 6.6 toont de drie cloud-modellen die via de HuggingFace Hub worden aangesproken. Deze komen qua basisarchitectuur overeen met de lokale varianten, maar worden aangeboden door externe inferentieproviders.

Tabel 6.7 toont de configuratie van EcoLogits en de inferentieparameters voor de cloud-aanroepen.

Parameter	Waarde
Meta-Llama 3.1 8B Instruct	meta-llama/Llama-3.1-8B-Instruct
Qwen3-VL-8B-Instruct	Qwen/Qwen3-VL-8B-Instruct
Apertus-8B-Instruct-2509	swiss-ai/Apertus-8B-Instruct-2509
Basis-URL	https://router.huggingface.co/v1

Tabel 6.6: Geëvalueerde cloud-modellen (HuggingFace Hub)

Parameter	Waarde
Geïnstrumenteerde provider	huggingface_hub
Initialisatiemoment	vóór import van huggingface_hub (module-niveau)
Energiescope	impacts.usage.energy (operationeel, excl. embodied carbon)
PUE	ingebouwd in EcoLogits-schatting
Prefill-correctiefactor α	0,175 (Caravaca e.a., 2025)
Inferentie temperature	0
Inferentie stream	True (met include_usage)
response_format	json_object

Tabel 6.7: EcoLogits- en inferentieparameters (cloud-inferentie)

```

1 from typing import List, Optional
2 from pydantic import BaseModel, Field
3
4 class Periode(BaseModel):
5     supplier: Optional[str] = None
6     start_date: Optional[str] = None    # Format: YYYY-MM-DD
7     end_date: Optional[str] = None     # Format: YYYY-MM-DD
8     kwh_quantity: Optional[float] = None
9
10 class FactuurModel(BaseModel):
11     """You are an expert at extracting structured data from Italian
12     utility bills (bollette). Extract the following fields and return
13     them as JSON.
14
15     CRITICAL RULES -- read carefully:
16     1. Only extract the INVOICED billing period -- the period this
17        specific invoice charges for.
18     2. IGNORE all historical tables such as "Andamento storico dei
19        prelievi", yearly overviews, and any table showing multiple
20        past months/years side by side.
21     3. A typical invoice covers exactly ONE billing period (one
22        month).
23     4. A date range spanning a full year is NEVER a valid billing
24        period -- it is always a historical summary.
25     5. The kwh_quantity MUST correspond to the same billing period as
26        start_date/end_date.
27
28     WHERE TO FIND EACH FIELD:
29     - supplier: the energy company name, top of first page.
30     - start_date / end_date: look in "importi riferiti al mese di",
31        "periodo di fornitura", "fornitura dal ... al ...", or
32        "dettagli riferiti alla fattura".
33     - kwh_quantity: total active energy in kWh for the invoiced
34        period.
35        If split by bands (F1/F2/F3), SUM those values from "Misure" or
36        "Dettaglio dei consumi".
37     If supplier, start_date, end_date or kwh_quantity is not present,
38     set it to null.
39     Always return exactly one entry in the periodes list."""
40
41     periodes: List[Periode] = Field(... , min_length=1)

```

Codefragment 6.1: Het definitieve Pydantic-extractieschema (`modello.py`): `FactuurModel` combineert JSON-schemavalidatie met Italiaanse semantische ankerpunten in de docstring.

```

1  # PUE niet meegegeven aan tracker: CodeCarbon past PUE toe op
    energy_consumed,
2  # waardoor raw_energy_kwh en energy_kwh_with_pue niet meer los te
    houden zijn.
3  # PUE wordt handmatig toegepast na het stoppen van de tracker.
4  tracker = EmissionsTracker(
5      project_name="local_inference",
6      measure_power_secs=0.5,
7      save_to_file=False,
8      log_level="warning",
9  )
10 tracker.start()
11 try:
12     response = requests.post(f"{base_url}/chat/completions",
13                             json=payload)
14     response.raise_for_status()
15 finally:
16     tracker.stop()
17
18 d = tracker.final_emissions_data
19 raw_energy_kwh = d.energy_consumed          # bare IT-load: GPU +
    CPU + RAM
20 energy_kwh_with_pue = raw_energy_kwh * PUE_LOCAL # facility
    overhead (PUE) verrekend
21 impacts = {
22     "energy_kwh": raw_energy_kwh,
23     "cpu_energy_kwh": d.cpu_energy,
24     "gpu_energy_kwh": d.gpu_energy,
25     "ram_energy_kwh": d.ram_energy,
26 }

```

Codefragment 6.2: Isolatie en registratie van hardwarematig energieverbruik via CodeCarbon in LlamaCppProvider.

```
1 from ecologits import EcoLogits
2 from ecologits.tracers.huggingface_tracer import llm_impacts
3
4 EcoLogits.init(providers=["huggingface_hub"])
5
6 # Modellen die niet in de EcoLogits-databank zitten, handmatig
  registreren
7 repo = llm_impacts.__globals__['models']
8 custom_models = [
9     {
10         'provider': 'huggingface_hub',
11         'name': 'qwen/qwen3-vl-8b-instruct',
12         'architecture': {'type': 'dense', 'parameters': 9}
13     },
14     {
15         'provider': 'huggingface_hub',
16         'name': 'meta-llama/llama-3.1-8b-instruct',
17         'architecture': {'type': 'dense', 'parameters': 8.03}
18     },
19     {
20         'provider': 'huggingface_hub',
21         'name': 'swiss-ai/Apertus-8B-Instruct-2509',
22         'architecture': {'type': 'dense', 'parameters': 8}
23     },
24 ]
25 for m in custom_models:
26     repo.add_model(m)
```

Codefragment 6.3: Manuele registratie van aangepaste modellen in de EcoLogits-modeldatabank, zodat de LCA-gebaseerde energieschatting ook voor niet-gecatalogeerde modellen beschikbaar is.

7

Simulatie en Dataverzameling

In dit hoofdstuk wordt beschreven hoe de gesimuleerde werklasten worden uitgevoerd en hoe de experimentele data wordt verzameld. De testomgeving die in 6 werd opgebouwd, vormt de basis voor de gecontroleerde uitvoering van de prestatietests. Per aanroep worden tijdstempels, responstijden, tokentellingen en energiemetingen geregistreerd, waarna de resultaten worden opgeslagen en gescored ten opzichte van de ground truth.

7.1. Orkestratie en Serverlevenscyclus

De `BenchmarkRunner` klasse orkestreert de uitvoering van de experimenten. Lokale modellen worden bediend via een extern `llama-server` subprocess, beheerd door de `LocalServerManager` klasse. Deze klasse start de server met de juiste `GGUF`-parameters, blokkeert op het `GET /health`-eindpunt totdat het model volledig geladen is in het `VRAM`, en beëindigt het proces na afloop om het `VRAM` volledig vrij te geven.

7.1.1. Cloud Inferentie: Document-eerst batchverwerking

Voor `LLM's` die via een cloud-API (zoals HuggingFace Inference API) worden aangeroepen, fungeert de applicatie als een stateless cliënt. Hier wordt een *document-eerst* batchverwerkingsstrategie toegepast: voor elk document worden de verschillende cloud-modellen sequentieel aangesproken via HTTP-verzoeken. Aangezien het beheer van het model in het `VRAM` volledig door de cloud-provider wordt geabstraheerd, veroorzaakt deze iteratieve aanpak geen lokale prestatieknelpunten of energie-overhead.

Alle PDF-documenten worden voor aanvang van de cloud-inferentie reeds geëxtraheerd naar Markdown en gecached. Dit garandeert dat de tijdsmeting per inferentie geen extractie-overhead bevat.

7.1.2. Lokale Inferentie: Serverlevenscyclus en Batching

Bij lokale inferentie is de context fundamenteel anders. Het inladen van een GGUF-model van opslag naar het GPU VRAM is een uiterst I/O-intensief en energiever-slindend proces. Het framework ondersteunt twee strategieën, instelbaar via de omgevingsvariabele `LLAMA_RELOAD_MODEL_PER_CALL`:

- **Per-aanroep isolatie** (`true`, standaard): Voor elk (`model`, `document`)-paar wordt een verse `llama-server` instantie gestart en na de inferentie gestopt. Dit garandeert een volledig schone VRAM-toestand per meting, ten koste van extra laadtijd per oproep.
- **Per-model batch** (`false`): Het model wordt éénmalig geladen voor de volledige documentbatch. Dit vermijdt herhaalde laadpieken maar veronderstelt dat de servercache geen kruisbesmetting veroorzaakt tussen documenten, wat gegarandeerd wordt doordat elke inferentieoproep een verse context opent (`temperature=0`, geen chatgeschiedenis).

Codefragment 7.1 toont de per-aanroep isolatie-strategie die in dit experiment werd gehanteerd (`LLAMA_RELOAD_MODEL_PER_CALL=true`). De buitenste lus itereert over lokale modellen; de binnenste lus itereert over documenten en extractors. Met `reload_per_call=True` wordt vóór elke (`model`, `pdf`, `extractor`)-combinatie een verse `llama-server`-instantie gestart en direct daarna gestopt. Dit garandeert een volledig schone VRAM-toestand per meting en elimineert cache-interferentie tussen opeenvolgende documenten, ten koste van extra laadtijd per aanroep.

7.2. Resultatenopslag

Elk inferentieresultaat wordt door de `ResultScorer` module tegelijk op twee manieren opgeslagen:

- **Per-run JSON**: een individueel bestand per oproep in `results/results_YYYYMMDD_HHMMSS/`, met de volledige `ExtractionResult`-dump inclusief geëxtraheerde data, timing, energiemetingen en eventuele foutberichten.
- **SQLite-databank**: een gedeelde `results/results.db` die over alle sessies heen beschikbaar blijft. Elke rij vertegenwoordigt één inferentieoproep en bevat kolommen voor de provider, het model, het document, alle tijdsmetingen (TTFT, generatietijd, totale inferentietijd), tokentellingen, energieverbruik per component (CPU, GPU, RAM) en de regionale CO₂eq-projecties.

De SQLite-structuur maakt het mogelijk om resultaten van meerdere testronden (beheerd via `run_benchmark_loop.py` met een `run_number` per iteratie) samen te voegen en via standaardtools zoals pandas of DuckDB te analyseren, zonder dat exportstappen vereist zijn.

```

1 server = LocalServerManager()
2
3 for llm_config in local_configs:
4     model_name = llm_config["model"]
5     model_path = resolve_local_model_path(model_name)
6
7     for pdf_file in pdf_files:
8         for extractor_name in extractors:
9             # Verse instantie per (model, pdf, extractor): schone
              VRAM-toestand
10            server.start(model_path)      # blokkeert tot /health OK
11            provider = LlamaCppProvider(server.base_url, model_name)
12
13            result = runner._run_extraction_with_provider(
14                pdf_path=pdf_file,
15                extractor_name=extractor_name,
16                provider=provider,
17                extraction_model=FactuurModel,
18            )
19            scorer.save_result(result, run_number=run_number)
20
21            server.stop()      # verwijdert model uit VRAM

```

Codefragment 7.1: Per-aanroep isolatie via LocalServerManager (LLAMA_RELOAD_MODEL_PER_CALL=true, standaard): het GGUF-model wordt per (model, pdf, extractor)-combinatie als extern serverproces gestart en direct daarna beëindigd, wat een volledig schone VRAM-toestand per meting garandeert.

7.3. Evaluatie en Scoring

Om de kwaliteit van de LLM-uitvoer te bepalen, bevat de applicatie een ResultScorer module met een regelgebaseerd scoringssysteem. Na elke extractie roept de verwerkingsketen de methode validate_and_score() aan, die de geëxtraheerde JSON structureel valideert volgens het 100-puntensysteem dat volledig is uitgewerkt in Sectie 5.2.5. De score wordt per inferentie-aanroep berekend en opgeslagen in de SQLite-databank. De ground truth-map (ground_truth/) bevat de referentiewaarden per document en kan achteraf worden gebruikt voor een inhoudelijke nauwkeurigheidsanalyse via de calculate_similarity()-methode, maar deze stap maakt geen deel uit van de geautomatiseerde testprocedure.

8

Resultaten / Analyse en interpretatie van data

In dit hoofdstuk worden de experimentele resultaten van de PoC-prestatietest geanalyseerd en geïnterpreteerd. De data zijn afkomstig uit 6241 inferentieaanroepen verdeeld over zes modellen (drie cloud, drie lokaal), 21 documenten en tot 50 herhaalde runs per model-documentcombinatie (zie Hoofdstuk 7 voor de meetopstelling). Er worden 59 inferentieaanroepen niet geanalyseerd omdat deze faalden door netwerkproblemen. De grafieken zijn gegenereerd met behulp van een reproduceerbaar Python-script (`charts_resultaten/generate_charts.py`) op basis van de centrale SQLite-databank (`results/results.db`).

Methodologische noot: bandbreedte van cloud-energieschattingen.

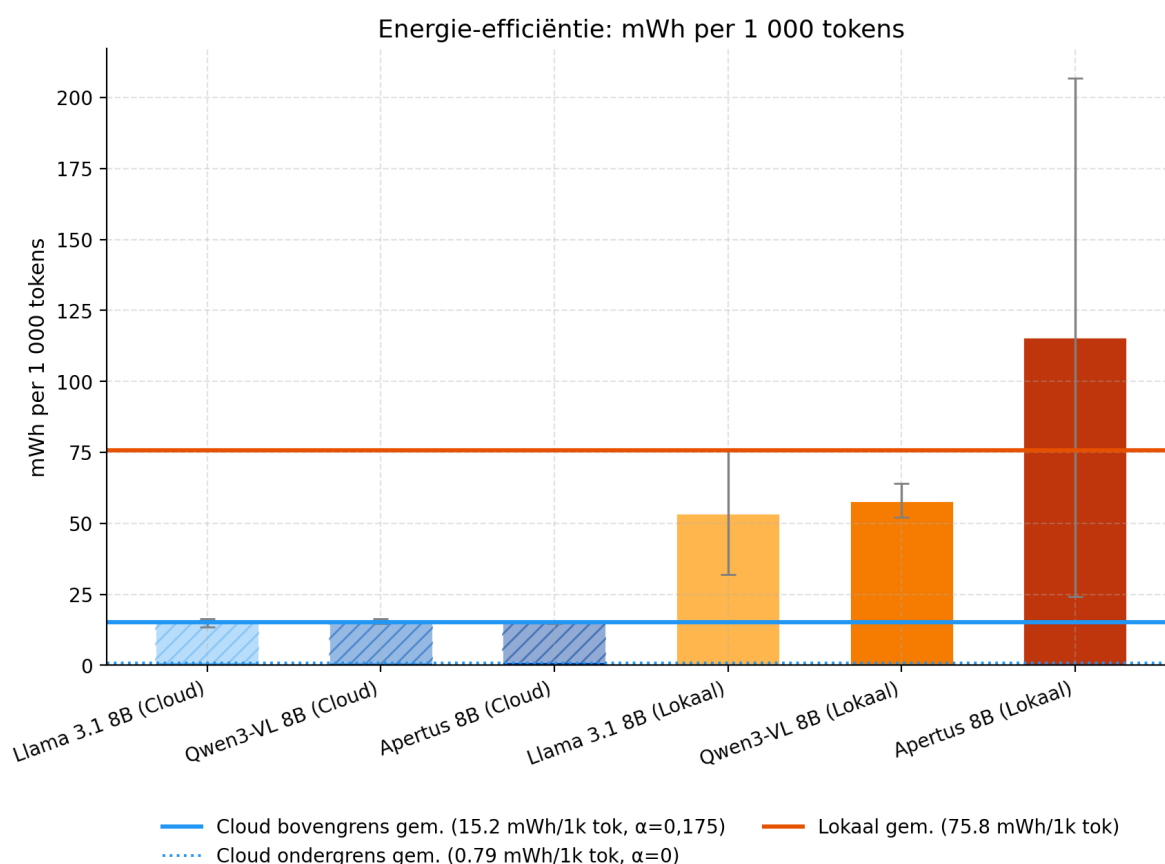
Alle cloud-energiewaarden in dit hoofdstuk zijn gebaseerd op het `energy_kwh_with_pue`-veld, dat de EcoLogits-basisschatting corrigeert voor de prefill-fase via de Caravacacorrectiefactor κ (zie Sectie 6.3.3). De **werkelijke cloud-energie bevindt zich in een bandbreedte**: de *ondergrens* is de onbewerkte EcoLogits-schatting (uitsluitend outputtokens, $\alpha = 0$); de *bovengrens* is de gecorrigeerde waarde ($\alpha = 0,175$). In de grafieken is deze bandbreedte voor cloud expliciet gevisualiseerd via een gearceerde zone op de cloud-balken: het gevulde gedeelte stelt de ondergrens voor, het gearceerde gedeelte ($\alpha = 0 \rightarrow 0,175$) het onzekere interval. De gemiddelde correctiefactor $\kappa_{\text{gem}} \approx 30,7$ impliceert dat de ondergrens gemiddeld **31x lager** ligt dan de bovengrens (zie Figuur 8.6). Door consistent de bovengrens te hanteren, is de analyse *conservatief ten voordele van de lokale opstelling*: indien de werkelijke cloud-energie dicht bij de ondergrens ligt, is het cloud-voordeel nog groter dan hier gerapporteerd.

Voor elke dimensie wordt de analyse opgezet als volgt: eerst worden de individuele modelresultaten besproken, daarna wordt het overkoepelende gemiddelde

voor cloud versus lokaal geëxtraheerd en geïnterpreteerd. Statistische significantie wordt beoordeeld via een tweezijdige Welch- t -toets (ongelijke varianties verondersteld), aangezien de standaardafwijkingen van beide groepen sterk verschillen. Bij $n > 3000$ per groep convergeert de t -verdeling naar de standaard normaalverdeling (meer dan 3000 vrijheidsgraden), waardoor de Welch- t -toets asymptotisch equivalent is aan een z -toets; beide procedures leveren in deze context nagenoeg identieke p -waarden op.

8.1. Energieverbruik

8.1.1. Energie-efficiëntie per 1000 tokens

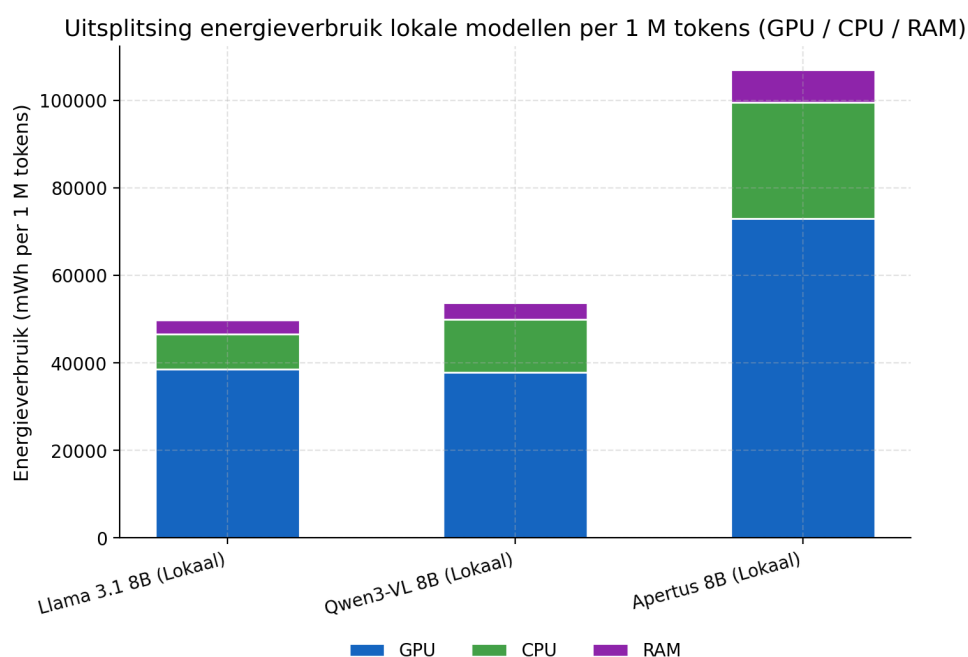


Figuur 8.1: Energie-efficiëntie per model (mWh per 1000 tokens), genormaliseerd op totaal tokenvolume (input + output). Gemiddelde cloud bovengrens: **15,2 mWh/1k tok** ($\alpha = 0,175$, $\sigma = 1,1$); cloud ondergrens: **≈0,50 mWh/1k tok** ($\alpha = 0$); lokaal gemiddelde: **75,8 mWh/1k tok** ($\sigma = 61,2$); gemiddelde verhouding lokaal/cloud: $\approx \times 5,0$. Gearceerde zone op cloud-balken: bandbreedte EcoLogits-ondergrens ($\alpha = 0$, enkel outputtokens) tot prefill-gecorrigeerde bovengrens ($\alpha = 0,175$, empirisch bepaald door (Caravaca e.a., 2025)). Foutbalken: $\pm 1 \sigma$.

Figuur 8.1 normaliseert het energieverbruik op het totale tokenvolume (input + output), wat de meest correcte maatstaf oplevert voor hardware-efficiëntie: aanroepen met lange documenten en aanroepen met korte documenten worden op de-

zelfde eenheid gebracht. De verhouding lokaal/cloud bedraagt $\approx 3,7$ tot $\approx 8,5$, afhankelijk van het model. Met name Apertus 8B lokaal (≈ 116 mWh/1k tok) scoort significant slechter dan de cloud-tegenhangers op bovengrens (≈ 15 mWh/1k tok; ondergrens $\approx 0,5$ mWh/1k tok), terwijl Llama 3.1 8B lokaal (≈ 54 mWh/1k tok) de relatief efficiëntste lokale variant is. Dit sluit aan bij de bevinding uit de literatuur (Sectie 2.2.1) dat kwantisatie het energieverbruik per token significant verlaagt ten opzichte van een niet-geoptimaliseerde basislijn. De experimentele resultaten tonen echter aan dat deze winst het verschil in hardware-efficiëntie ten opzichte van enterprise-GPU's niet volledig overbruggt. Het verschil is statistisch uiterst significant (Welch-t-test: $t = -42,5$, $p < 10^{-300}$).

8.1.2. Uitsplitsing lokale energie: GPU, CPU en RAM

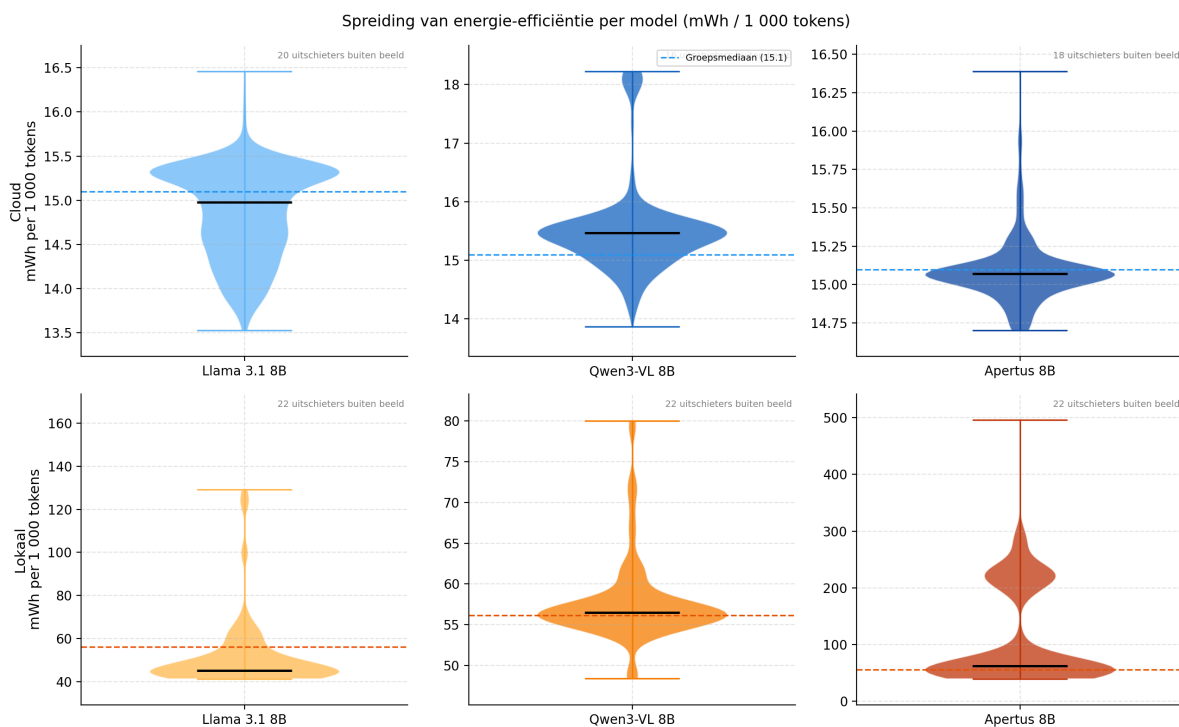


Figuur 8.2: Gemiddeld energieverbruik van lokale modellen per 1M tokens, uitgesplitst naar hardwarecomponent (GPU, CPU, RAM). Gemeten via CodeCarbon (Courty e.a., 2026); normalisatie op tokenvolume maakt modellen vergelijkbaar ongeacht documentlengte. De GPU-component is bij alle drie modellen goed voor meer dan 85% van het totale verbruik.

Figuur 8.2 normaliseert het componentverbruik op tokenvolume, waardoor de hardware-efficiëntie van de drie modellen direct vergelijkbaar is los van eventuele verschillen in gemiddelde documentlengte. De GPU is bij alle drie lokale modellen de dominante energieconsument, goed voor meer dan 85% van het totale verbruik. Dit is consistent met de architectuur van de testomgeving: Llama-server voert alle matrixvermenigvuldigingen op de GPU uit en de CPU beheert uitsluitend de orchestratielogica. Apertus 8B verbruikt per 1M tokens significant meer GPU-energie dan Llama 3.1 8B, wat bevestigt dat zijn hogere energie-per-

token-getal (zie Figuur 8.1) een inherent modelkenmerk is en geen artefact van langere documenten. De CPU-bijdrage is beperkt en relatief stabiel tussen modellen; RAM-verbruik is verwaarloosbaar klein dankzij de DIMM-heuristiek van CodeCarbon.

8.1.3. Spreiding van het energieverbruik



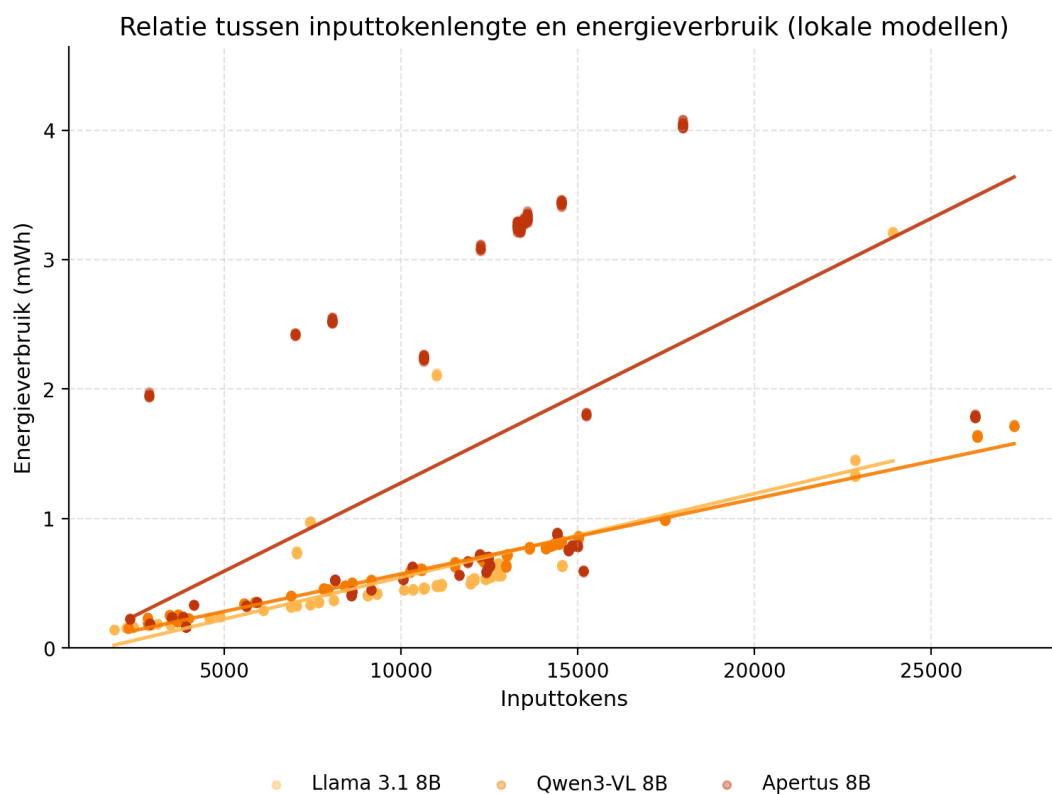
Figuur 8.3: Spreiding van de energie-efficiëntie per model (mWh per 1000 tokens) over alle runs en documenten, weergegeven als violinplot in zes afzonderlijke panelen met eigen y-as. Y-as per paneel begrensd tot het 1e–99e percentiel $\pm 10\%$ marge; het aantal extreme uitschieters buiten beeld wordt rechtsboven per paneel vermeld. Groepsmediaan cloud: ≈ 15 mWh/1k tok; groepsmediaan lokaal: ≈ 55 mWh/1k tok. De smalle cloud-verdeling is methodologisch verwacht: de EcoLogits-schatting schaalt lineair met het aantal outputtokens bij vaste modelgrootte en batchgrootte (GenAI Impact, [g.d.-a](#)).

Figuur 8.3 toont de spreiding per model in zes afzonderlijke panelen, opgesplitst naar cloud (bovenrij) en lokaal (onderrij). Omdat de absolute schalen sterk verschillen — cloud-modellen clusteren rond ≈ 15 mWh/1k tok, terwijl lokale modellen variëren van ≈ 50 tot meer dan 200 mWh/1k tok — heeft elk paneel een eigen y-as. Zo wordt de interne spreiding van elk model zichtbaar zonder dat uitschieters van één model de rest onderdrukken.

Cloud-modellen vertonen een smalle, geconcentreerde verdeling: de vioolomtrek is smal en de mediaan ligt dicht bij de groepsmediaan. Dit is methodologisch verwacht: de EcoLogits-schatting schaalt lineair met het aantal outputtokens bij vaste modelgrootte en batchgrootte (GenAI Impact, [g.d.-a](#)), waardoor variabiliteit uitslui-

tend voortkomt uit verschillen in outputtokenvolume. Bij lokale modellen is de verdeling structureel breder; CodeCarbon registreert ook pieken in GPU-belasting, thermische throttling en andere systeemeffecten. Apertus 8B valt op met een bijzonder brede en asymmetrische verdeling, wat wijst op grotere inconsistentie in energieverbruik per document.

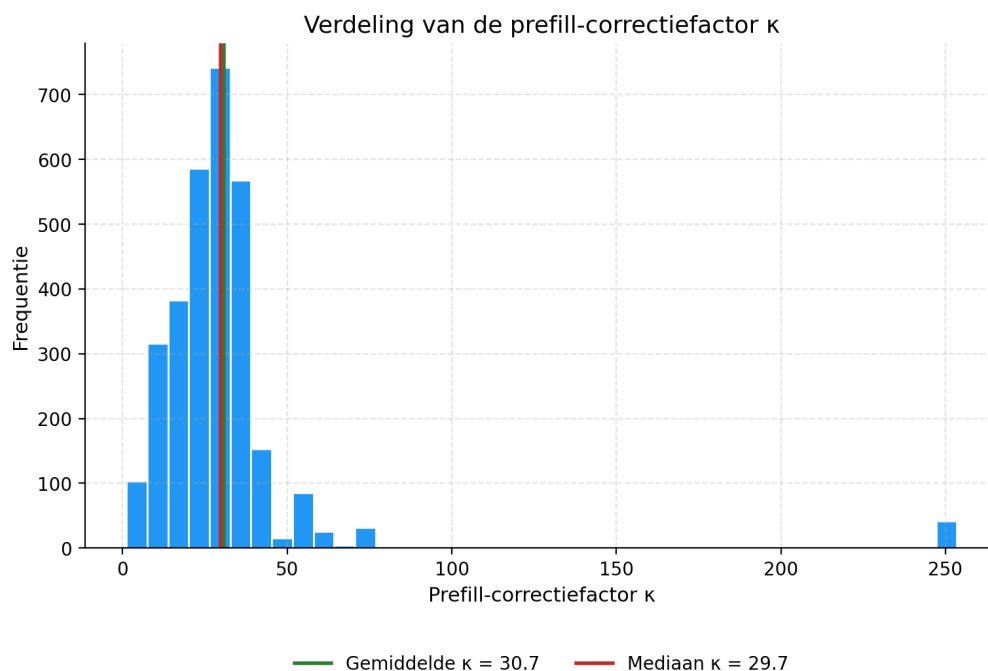
8.1.4. Relatie tussen documentlengte en energieverbruik



Figuur 8.4: Inputtokenlengte versus energieverbruik per aanroep (mWh) voor de drie lokale modellen, met OLS-regressielijnen per model. Gemeten via CodeCarbon (Courty e.a., 2026). Y-as begrensd op het 99e percentiel; uitschieters buiten beeld. De steilere helling van Apertus 8B ten opzichte van Llama 3.1 8B reflecteert zijn hoger energie-per-token-getal (zie ook Figuur 8.1).

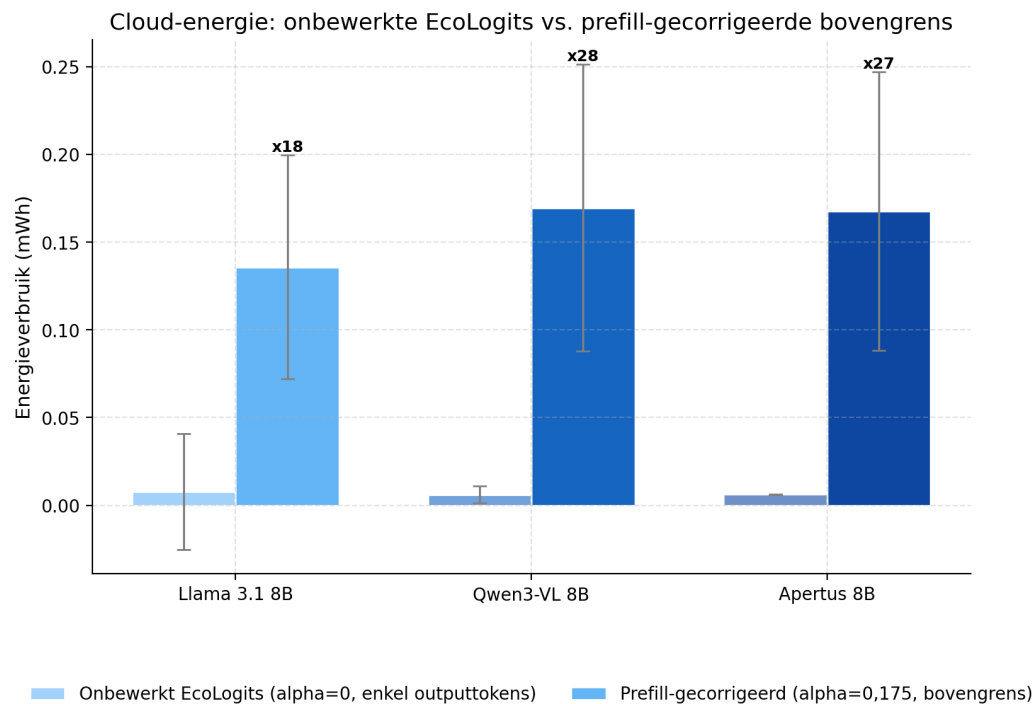
Figuur 8.4 illustreert de sterke lineaire relatie tussen de lengte van het input-document (na preprocessing naar Markdown) en het energieverbruik bij lokale inferentie. De regressielijnen per model zijn duidelijk onderscheidbaar: Apertus 8B heeft een steilere helling dan Llama 3.1 8B, wat aansluit bij zijn hogere energie-per-token-getal. Deze bevinding sluit aan bij het theoretische kader in Sectie 2.2, dat het onderscheid tussen prefill- en decodefase en het belang van de input/output-verhouding voor het energieprofiel beschrijft: bij dit belastingsprofiel (hoge input/output-verhouding) blijkt de prefill-fase het energieverbruik te domineren, en schaalt die fase lineair met de inputlengte.

8.2. Prefill-correctiefactor en impact op cloud-energie



Figuur 8.5: Frequentieverdeling van de EcoLogits-prefill-correctiefactor κ over alle cloud-aanroepen. Gemiddelde $\kappa = 30,7$ (mediaan: 29,7; min: 1,1; max: 253). κ compenseert het feit dat EcoLogits enkel de decode-fase schat; de correctie schaal met $\alpha \times (\text{inputtokens/outputtokens})$, waarbij $\alpha = 0,175$ empirisch is bepaald door (Caravaca e.a., 2025). De brede staart ($\kappa > 100$) correspondeert met de langste documenten in de dataset.

Figuur 8.5 toont de verdeling van de prefill-correctiefactor κ (zie Sectie 6.3.3 voor de formule). De gemiddelde κ bedraagt **30,7** (mediaan: 29,7; min: 1,1; max: 253). Dit betekent dat de onbewerkte EcoLogits-schatting (die uitsluitend de decode-fase meeneemt) gemiddeld een factor 31 *te laag* is voor het documentextractie-toepassingsscenario van deze bachelorproef. Zonder deze correctie zou de cloud-infrastructuur ten onrechte als extreem energiezuinig worden gepresenteerd. De brede staart van de verdeling ($\kappa > 100$) correspondeert met de langste documenten in de dataset (elektriciteits- en waterfacturen met meer dan 20.000 input-tokens), zoals de scatter-analyse in Figuur 8.4 bevestigt.

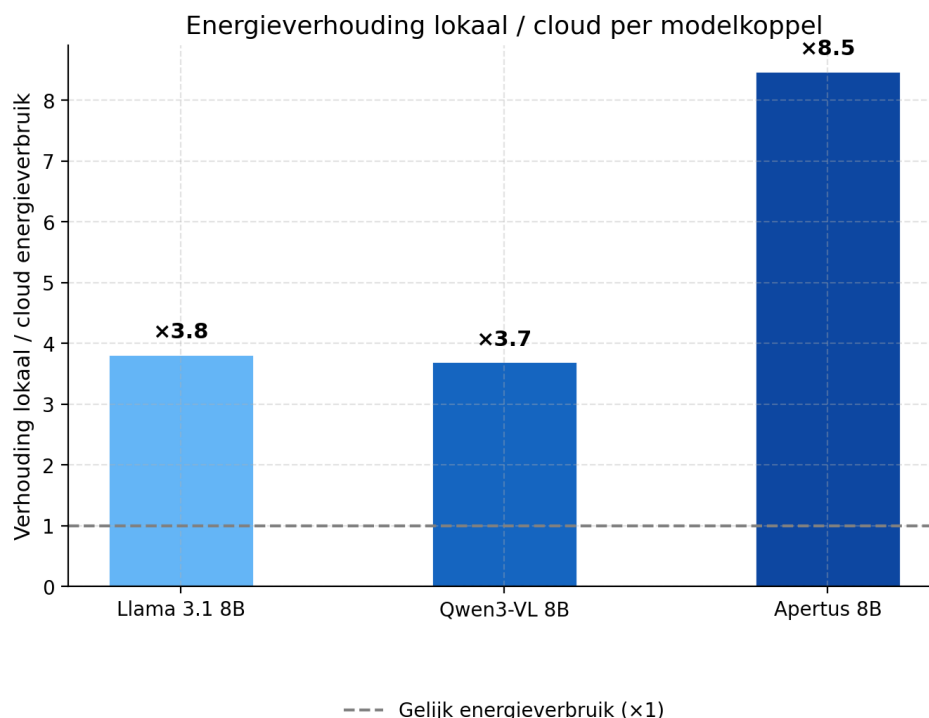


Figuur 8.6: Cloud-energieverbruik per model (mWh per aanroep): onbewerkte EcoLogits-schatting ($\alpha = 0$, uitsluitend outputtokens, $\approx 0,005\text{--}0,006$ mWh per aanroep) versus prefill-gecorrigeerde bovengrens ($\alpha = 0,175$, $\approx 0,150\text{--}0,185$ mWh per aanroep). Annotaties tonen de correctiefactor κ per model ($\times 30\text{--}31$). $\alpha = 0,175$ empirisch bepaald door (Caravaca e.a., 2025); EcoLogits-methodologie: (Rincé & Banse, 2025b). Foutbalken: $\pm 1\sigma$.

Figuur 8.6 maakt de impact van de correctie concreet zichtbaar. De onbewerkte EcoLogits-waarden ($\approx 0,005\text{--}0,006$ mWh) zijn gemiddeld een factor 30–31 lager dan de gecorrigeerde waarden ($\approx 0,150\text{--}0,185$ mWh). Dit enorme verschil is specifiek voor het belastingsprofiel van dit experiment: bij een verhouding van $\approx 150:1$ inputtokens tot outputtokens domineert de prefill-fase volledig het energieverbruik, maar EcoLogits registreert die fase niet (Rincé & Banse, 2025a).

Consequentie voor de vergelijking: alle cloud-energiewaarden in dit hoofdstuk zijn gebaseerd op de *gecorrigeerde bovengrens* (*energy_kwh_with_pue*). Omdat de werkelijke energie bevindt zich tussen de onbewerkte waarde en de theoretische bovengrens bij $\alpha = 1$, is de gehanteerde correctie ($\alpha = 0,175$, empirisch bepaald door Caravaca e.a. (2025)) de best beschikbare benadering. Door consequent de bovengrens te hanteren, is de vergelijking *conservatief ten voordele van de lokale opstelling*: mocht de werkelijke cloud-prefill-energie lager liggen dan $\alpha = 0,175$ suggereert, dan is het cloud-voordeel groter dan hier gerapporteerd.

8.3. Energie-verhouding per modelkoppel



Figuur 8.7: Verhouding lokaal/cloud energieverbruik per modelkoppel (berekend op gemiddeld `energy_kwh_with_pue` per aanroep). Llama 3.1 8B: $\approx \times 3,8$; Qwen3-VL 8B: $\approx \times 3,7$; Apertus 8B: $\approx \times 8,5$. Cloud-modellen draaien op FP16 op enterprise-GPU's (Hugging Face, 2026); lokale modellen zijn Q4_K_M-gekwantiseerd op een NVIDIA RTX 3070.

Figuur 8.7 brengt de energieverhouding per modelkoppel in kaart. De verhouding varieert van $\approx \times 3,7$ (Qwen3-VL 8B) tot $\approx \times 8,5$ (Apertus 8B), met Llama 3.1 8B op $\approx \times 3,8$. De lagere ratio voor Llama en Qwen wijst op een relatief efficiënte kwantisatie-implementatie van dit model; de hoge ratio voor Apertus reflecteert zijn tragere inferentie op de RTX 3070.

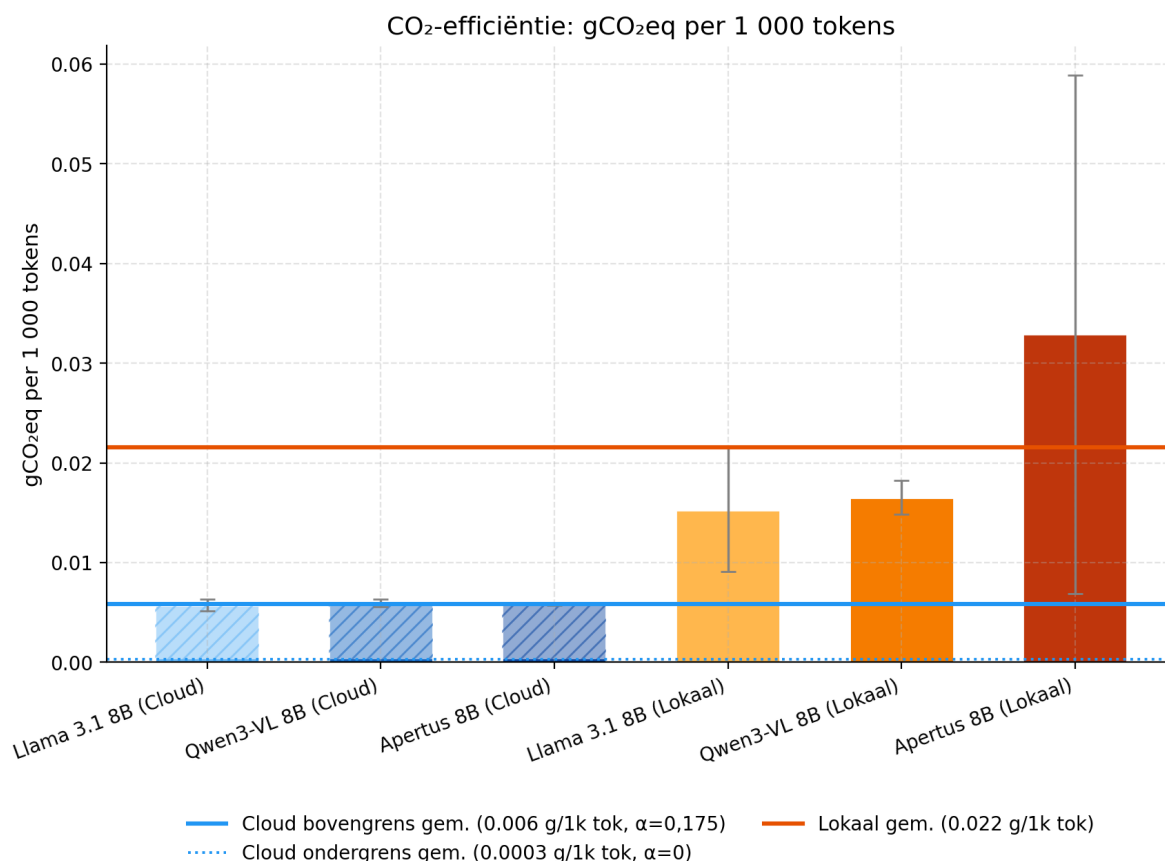
Het is methodologisch van belang te benadrukken (zie ook Sectie 5.1) dat dit geen *hardware-neutrale* vergelijking is: de cloud-versies draaien op FP16 op enterprise GPU's, terwijl de lokale versies Q4_K_M-gekwantiseerd zijn op een consumenten-GPU. De ratio weerspiegelt daarmee de *praktische realiteitsvergelijking* (cloud API versus geoptimaliseerde lokale edge-deployment) eerder dan een zuivere hardware-efficiëntiemeting.

8.4. CO₂-uitstoot

Omdat CO₂-uitstoot rechtstreeks evenredig is met het energieverbruik vermenigvuldigd met een emissiefactor, weerspiegelen de CO₂-resultaten het energiepatroon volledig. Genormaliseerd op tokenvolume bedraagt het cloud-gemiddelde **$\approx 5,8 \text{ mgCO}_2\text{eq per 1000 tokens}$** (EcoLogits usage-schatting, bovengrens $\alpha = 0,175$),

tegenover **≈21,6 mgCO₂eq per 1000 tokens** voor lokaal (CodeCarbon × Italiaanse emissiefactor 0,285 kgCO₂eq/kWh). De verhouding (≈ ×3,7) is vergelijkbaar met de energieverhouding.

8.4.1. CO₂-efficiëntie per 1000 tokens

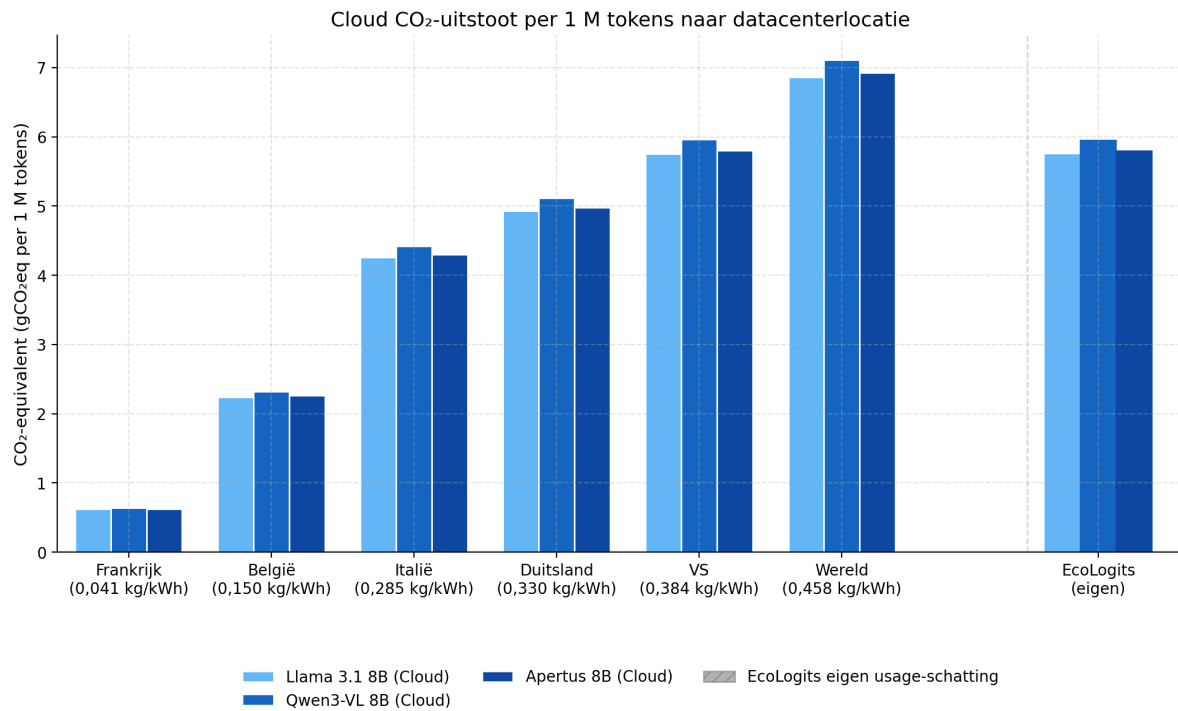


Figuur 8.8: CO₂-efficiëntie per model (gCO₂eq per 1000 tokens), genormaliseerd op tokenvolume. Gemiddelde cloud bovengrens: **0,0058 g/1k tok** ($\alpha = 0,175$); lokaal gemiddelde: **0,0216 g/1k tok**; verhouding ≈ ×3,7. Gearceerde zone: EcoLogits-bandbreedte ($\alpha = 0$ tot $\alpha = 0,175$; (Caravaca e.a., 2025)). Lokale emissiefactor Italië: 0,285 kgCO₂eq/kWh (Ember, 2026). Foutbalken: ±1 σ .

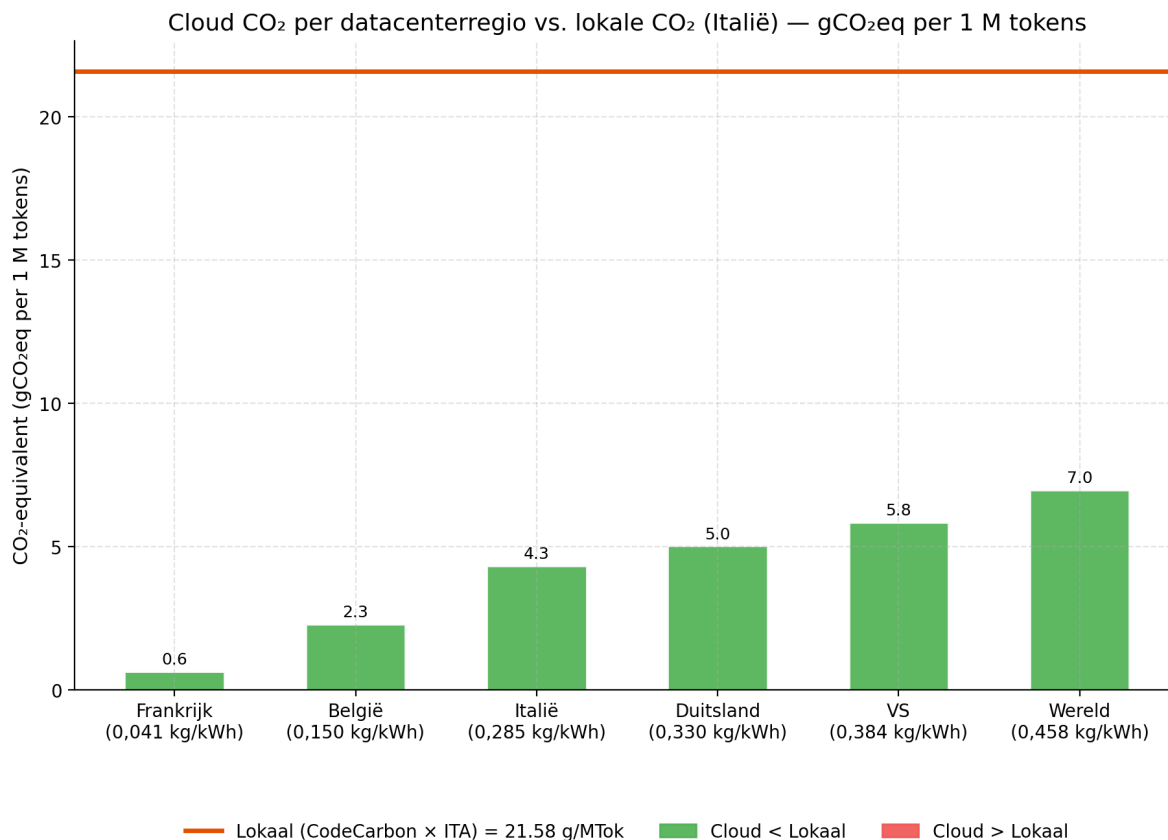
Figuur 8.8 toont de CO₂-efficiëntie per 1000 tokens, analoog aan de energiefiguur (Figuur 8.1). De relatieve verhoudingen volgen het energiepatroon: cloud is efficiënter dankzij enterprise-hardware en een groenere Europese stroomportfolio, terwijl lokale inferentie op een consumentenhardware in Italië een hogere CO₂-intensiteit per token kent.

8.4.2. Regionale gevoeligheidsanalyse: effect van datacenterlocatie

De regionale gevoeligheidsanalyse (Figuren 8.9 en 8.10) is inhoudelijk rijker: de *locatieafhankelijkheid* van de cloud-emissies is groter dan het verschil tussen cloud en lokaal zelf.



Figuur 8.9: Cloud CO₂-uitstoot per 1 M tokens naar datacenterlocatie, voor de drie cloud-modellen. Gearceerde balken: EcoLogits eigen usage-schatting (excl. embodied carbon). Berekend als $\text{energy_kwh_with_pue} \times \text{regionale emissiefactor} \times 10^9 / \text{total_tokens}$. Emissiefactoren (Ember, 2026): Frankrijk 0,041; België 0,150; Italië 0,285; Duitsland 0,330; VS 0,384; Wereld 0,458 kgCO₂eq/kWh.

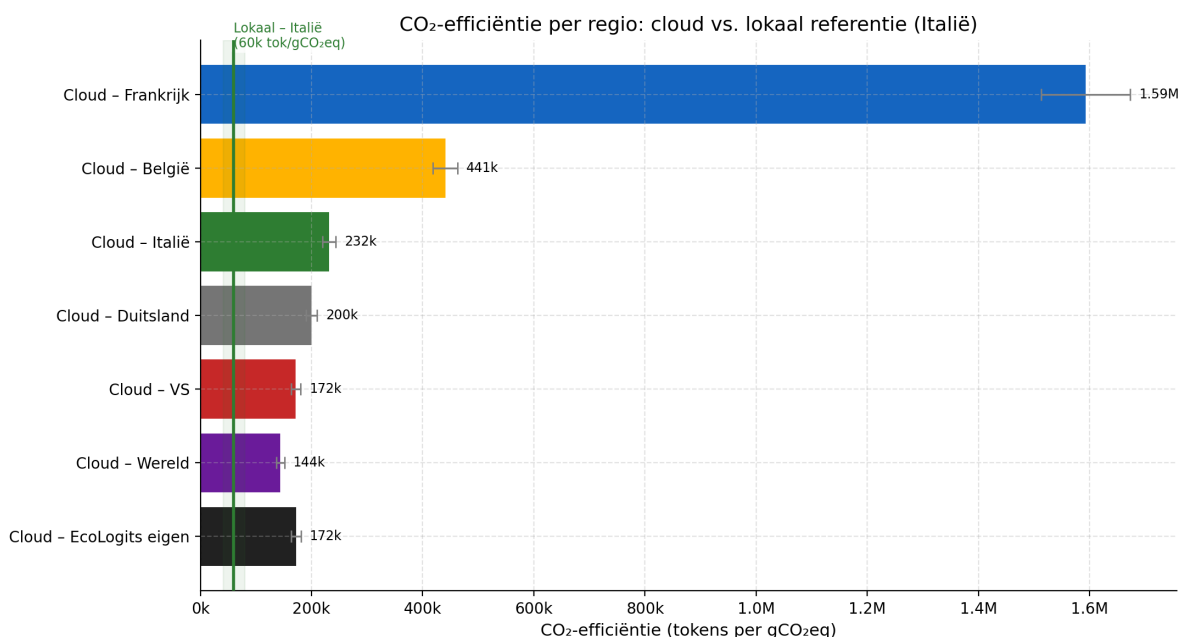


Figuur 8.10: Cloud CO₂ per datacenterregio (berekend als gemeten energie inclusief PUE-correctie × regionale emissiefactor, genormaliseerd op 1M tokens) versus de lokale referentiewaarde voor Italië (oranje lijn, 21,58 gCO₂eq/MTok). Groene balken: cloud gunstiger dan lokaal; rode balken: cloud ongunstiger. Waarden in gCO₂eq per 1M tokens. Variatie over regio's: factor ≈11 (van ≈0,9 g/MTok in Frankrijk tot ≈9,7 g/MTok wereldgemiddelde). Emissiefactoren per regio: (Ember, 2026); lokale meting via CodeCarbon (Courty e.a., 2026).

Figuur 8.10 toont per datacenterregio de berekende cloud CO₂-uitstoot ($\text{energy_kwh_with_pue} \times \text{regionale emissiefactor}$, genormaliseerd op 1M tokens) afgezet tegen de gemeten lokale waarde in Italië (oranje lijn, 21,58 gCO₂eq/MTok). In alle zes geanalyseerde regio's valt de cloud-uitstoot ruimschoots onder de lokale waarde — ook in de koolstofintensievere regio's zoals de VS en het wereldgemiddelde. Dit is ook de reden dat er geen rode balken zichtbaar zijn op Figuur 8.10: in geen enkele regio is de cloud-uitstoot hoger dan lokaal.

De variatie over de geanalyseerde regio's bedraagt een factor ≈11 (van ≈0,9 gCO₂eq/MTok in Frankrijk tot ≈9,7 gCO₂eq/MTok voor het wereldgemiddelde). Dit bevestigt de bevinding uit Sectie 2.4: de keuze van de cloudregio heeft een grotere invloed op de CO₂-uitstoot dan de keuze tussen lokaal en cloud zelf. Voor de casus van Turtle Srl, waarvan de infrastructuur in Italië staat (emissiefactor 0,285 kgCO₂eq/kWh), geldt dat cloud-inferentie via een datacenter in een land met een groenere stroommix ecologisch gunstiger uitvalt dan de lokale opstelling.

8.4.3. CO₂-efficiëntie per regio



Figuur 8.11: CO₂-efficiëntie (tokens per gCO₂eq) per cloud-regio, gesorteerd van schoonste naar vuilst stroomgrid. Waarden: Frankrijk ≈1,13 M; België ≈313 k; Italië ≈165 k; Duitsland, VS en wereld ≈102–142 k tok/gCO₂eq. De verticale lijn toont de lokale referentiewaarde (Italië, CodeCarbon (Courty e.a., 2026); ≈60 k tok/gCO₂eq). Foutbalken en gekleurde band: ±1σ. De EcoLogits-balk toont de eigen usage-gebaseerde CO₂-schatting (excl. embodied carbon, Europese gemiddelde emissiefactor). Emissiefactoren per regio: (Ember, 2026).

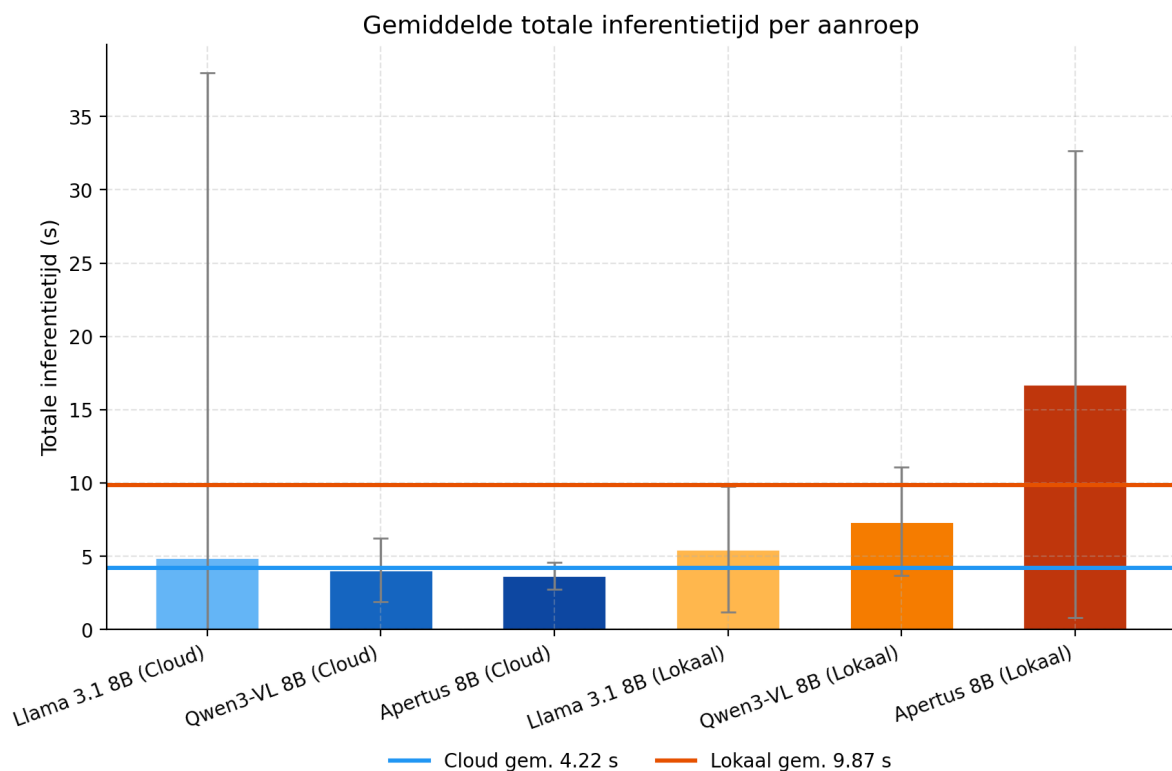
Figuur 8.11 vergelijkt de CO₂-efficiëntie van cloud-inferentie per regio met de gemeten lokale waarde in Italië (volle lijn, ≈60 k tok/gCO₂eq). Deze waarde is berekend als het gemiddelde van de per-aanroep verhouding tokens/gCO₂; door Jensen's ongelijkheid is dit niet de directe inverse van de 21,58 gCO₂eq/MTok uit Figuur 8.10, die gebruik maakt van het gemiddelde van gCO₂/tokens per aanroep.

Cloud-regio's met een groen stroomgrid scoren sterk beter dan lokaal. Frankrijk (nucleair grid, ≈1,13 M tok/gCO₂eq) en België (mix nucleair/hernieuwbaar, ≈313 k tok/gCO₂eq) overtreffen de lokale opstelling respectievelijk met een factor 19 en 5. Cloud-Italië (≈165 k tok/gCO₂eq) scoort beter dan lokaal dankzij de hogere energie-efficiëntie van het datacenter (PUE-voordeel). Regio's met een hogere koolstofintensiteit (Duitsland, VS, wereldgemiddelde: 102–142 k tok/gCO₂eq) presteren nog steeds beter dan de lokale meting, consistent met de regionale gevoeligheidsanalyse in Figuur 8.10.

De EcoLogits-balk (≈122 k tok/gCO₂eq) hanteert de eigen usage-gebaseerde CO₂-schatting van EcoLogits met een ingebouwde Europese gemiddelde emissiefactor, onafhankelijk van de gekozen regio; embodied carbon is niet inbegrepen. De doorvoercijfers (tokens per uur) worden apart besproken in Sectie 8.5.

8.5. Latentie en doorvoer

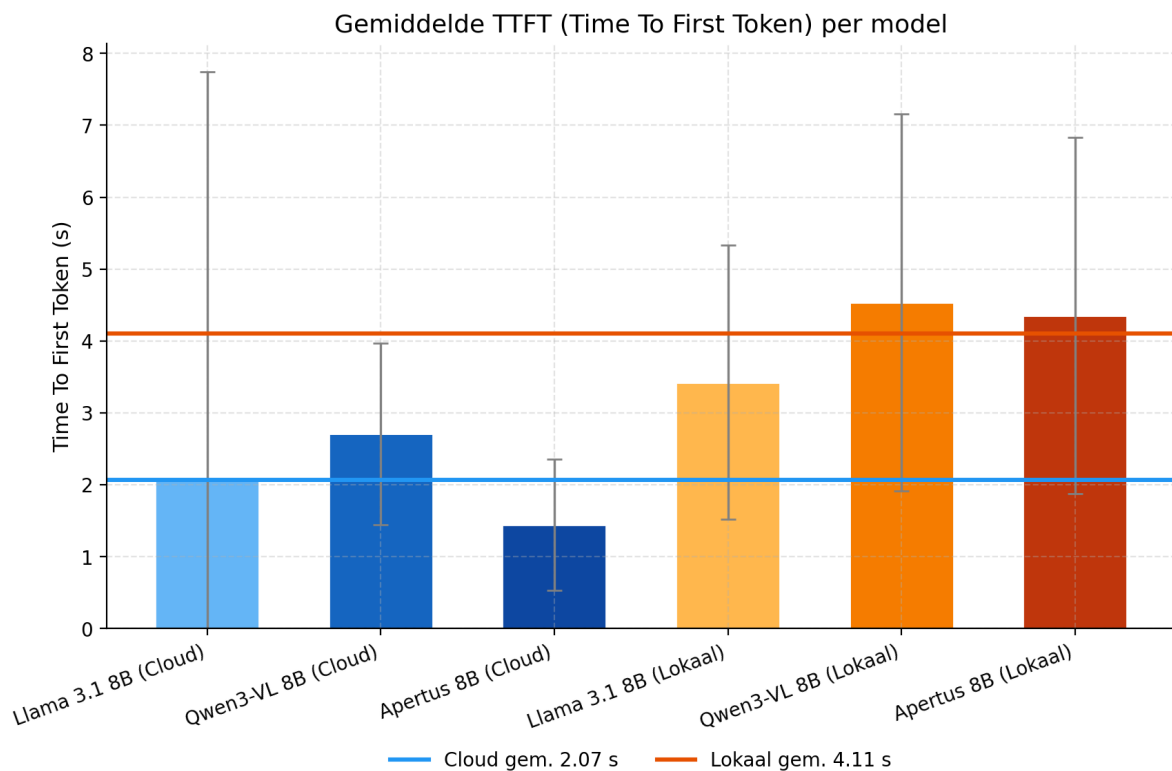
8.5.1. Totale inferentietijd



Figuur 8.12: Gemiddelde totale inferentietijd per aanroep per model, met foutbalken ($\pm 1\sigma$; ondergrens begrensd op 0). Gemiddeld cloud: **4,22 s**; gemiddeld lokaal: **9,87 s** ($\approx \times 2,3$ trager). Apertus 8B lokaal is een uitschieter met $\approx 16,7$ s gemiddeld door kwantisatie-eigenschappen van Q4_K_M op de RTX 3070.

Figuur 8.12 toont het gemiddelde van de totale inferentietijd. De cloud-modellen zijn gemiddeld **4,22 seconden** per aanroep, de lokale modellen **9,87 seconden**. Dit is een factor $\approx 2,3$ trager voor lokaal. Het grootste verschil betreft **Apertus 8B lokaal** ($\approx 16,7$ s gemiddeld), wat samenhangt met het kwantisatie-artefact van dit model: de Q4_K_M-versie van Apertus verwerkt tokens aanzienlijk langzamer dan de Llama-variant. Deze bevinding heeft praktische relevantie voor interactieve toepassingen: een wachttijd van >15 seconden wordt algemeen als onacceptabel beschouwd voor een responsieve gebruikersinterface.

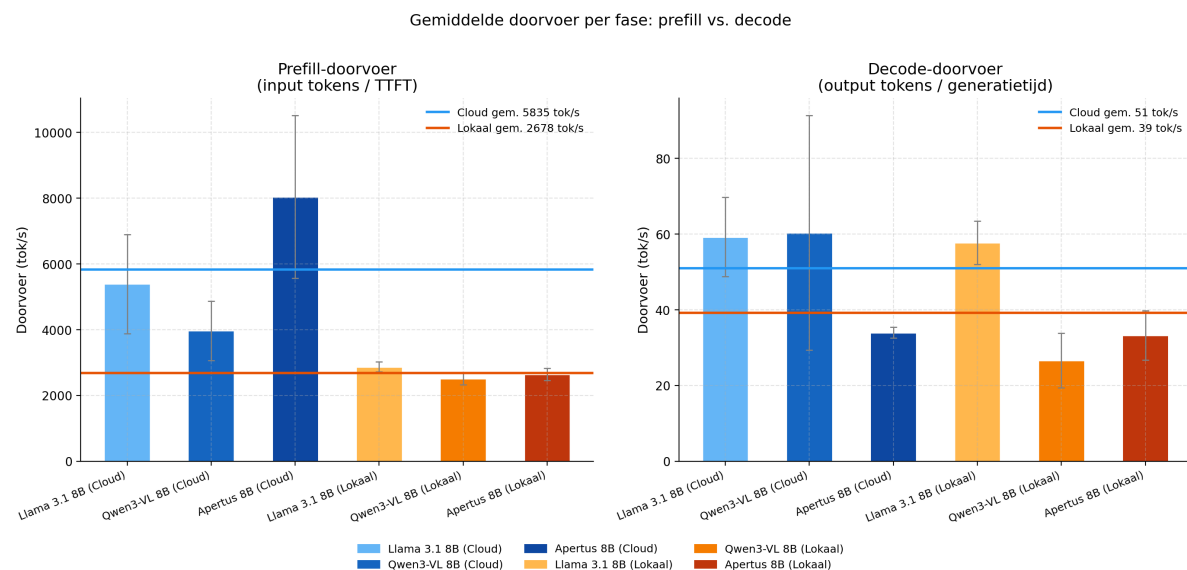
8.5.2. Time To First Token (TTFT)



Figuur 8.13: Gemiddelde TTFT per model, met foutbalken ($\pm 1\sigma$; ondergrens begrensd op 0). Cloud-TTFT is sterk beïnvloed door netwerklatentie en load-balancer-wachtrijen. Lokale TTFT ($\approx 3,4\text{--}4,5\text{ s}$) is relatief hoog door de grote documentprompts (gemiddeld $\approx 10\,000$ inputtokens) die de GPU zwaar belasten tijdens de prefill-fase.

Figuur 8.13 vergelijkt de TTFT. Zoals vermeld in Sectie 5.2.5 is de cloud-TTFT sterk beïnvloed door netwerklatentie en load-balancer wachtrijen, waardoor deze meting geen zuivere hardware-vergelijking is. Wat opvalt is dat de TTFT van lokale modellen ($\approx 3,4\text{--}4,5\text{ s}$) hoger ligt dan verwacht voor een lokale opstelling: dit is inherent aan de grote documentprompts (gemiddeld ≈ 10.000 inputtokens), die de GPU zwaar belasten tijdens de prefill-fase. De cloud-TTFT profiteert in dat opzicht van het parallelle verwerkingsparadigma op de grote GPU-clusters van de provider.

8.5.3. Doorvoer (tokens per seconde)



Figuur 8.14: Gemiddelde doorvoer per model, opgesplitst naar prefill-fase (inputtokens/TTFT, links) en decode-fase (outputtokens/generatietijd, rechts). Prefill gem.: cloud **5.835 tok/s** vs. lokaal **2.678 tok/s** ($\approx \times 2,2$). Decode gem.: cloud **51 tok/s** vs. lokaal **39 tok/s** ($\approx \times 1,3$). Foutbalken: $\pm 1 \sigma$; ondergrens begrensd op 0.

Figuur 8.14 splitst de doorvoer op in twee afzonderlijke fasen, wat een genuanceerder beeld geeft dan de eerder gerapporteerde totaalwaarde.

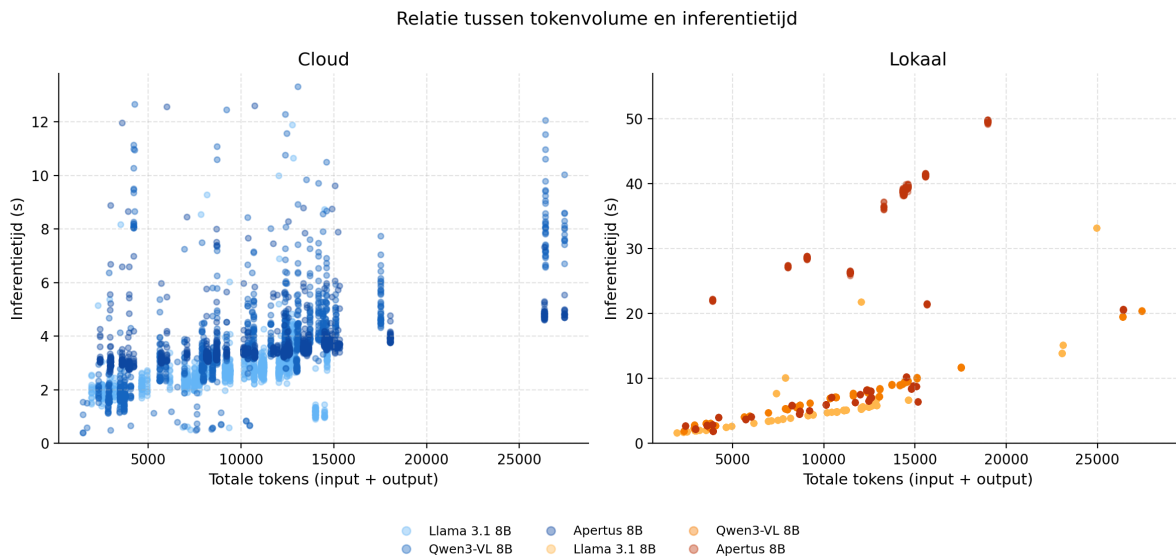
Prefill-doorvoer (inputtokens / TTFT): cloud-modellen verwerken gemiddeld **5.835 tok/s** in de prefill-fase, tegenover **2.678 tok/s** voor lokaal — een factor $\approx 2,2$. Dit verschil reflecteert de hogere parallelisme-capaciteit van enterprise-GPU's (H100/A100) bij het gelijktijdig verwerken van grote prompts. Apertus 8B cloud steekt er met ≈ 8.048 tok/s ver bovenuit; de drie lokale modellen zijn onderling vergelijkbaar (≈ 2.500 – 2.900 tok/s).

Decode-doorvoer (outputtokens / generatietijd): het verschil is hier aanzienlijk kleiner. Cloud haalt gemiddeld **51 tok/s**, lokaal **39 tok/s** — een factor $\approx 1,3$. De token-gewijze generatie is inherent serieel en schaalt minder met de hoeveelheid GPU-rekenkracht. Llama 3.1 8B (cloud: 59 tok/s; lokaal: 58 tok/s) toont zelfs nagenoeg gelijke decode-snelheid, wat suggereert dat de Q4_K_M-kwantisatie voor dit model de decode-fase nauwelijks benadeelt.

Implicatie: de eerder gerapporteerde totale doorvoer van ≈ 3.182 tok/s (cloud) versus ≈ 1.521 tok/s (lokaal) werd gedomineerd door de prefill-fase, die immers het grootste aandeel van de tokens vertegenwoordigt (gemiddeld $\approx 150:1$ input/output-verhouding). De decode-snelheid — die bepalend is voor de *zichtbare* generatiesnelheid in een interactieve interface — is tussen cloud en lokaal bijna vergelijkbaar. Voor de batchverwerking van Turtle Srl (niet-interactief) geldt dat de prefill-snelheid het knelpunt vormt; de cloud-architectuur biedt hier een

duidelijker voordeel.

8.5.4. Relatie tokenvolume en inferentietijd

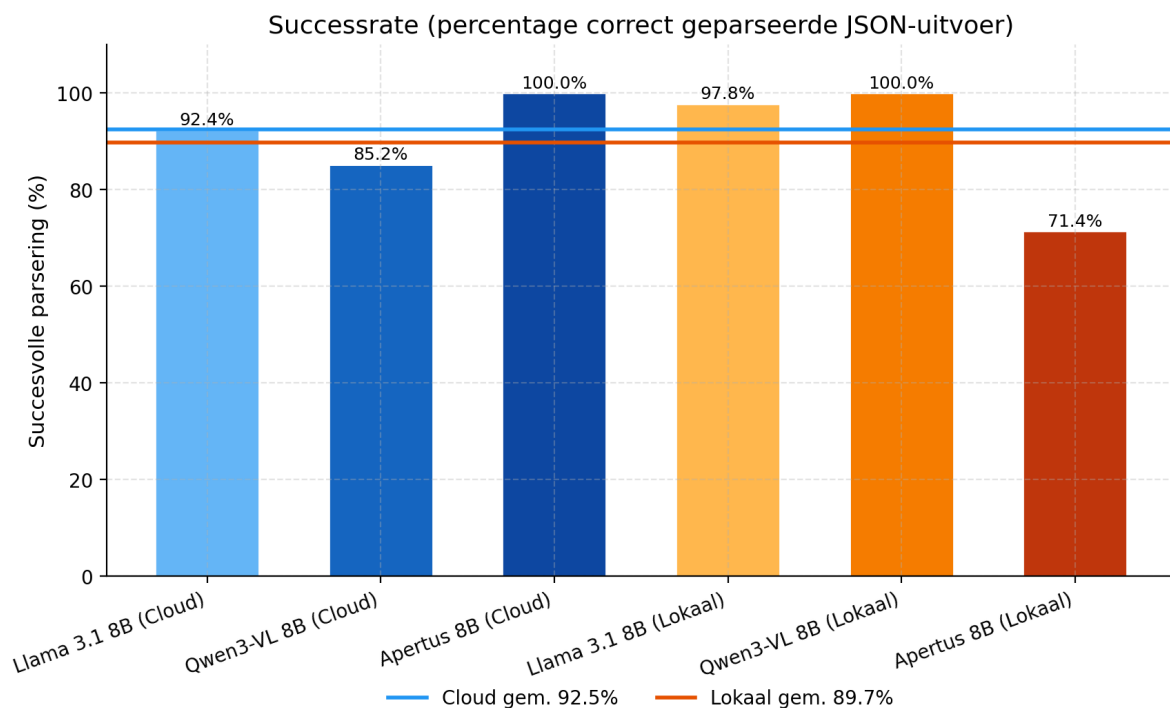


Figuur 8.15: Totaal tokenvolume versus inferentietijd per aanroep, gesplitst naar cloud (links) en lokaal (rechts). Y-as per paneel begrensd op het 99e percentiel; uitschieters buiten beeld. Lokale inferentie toont een sterk lineair verband met weinig scatter; cloud-metingen zijn spreidender door variabele netwerk- en provider-latentie.

Figuur 8.15 toont dat bij lokale inferentie een sterk lineair verband bestaat tussen tokenvolume en inferentietijd, met weinig scatter. Bij cloud-modellen is de spreiding groter, wat consistent is met de variabiliteit van netwerk- en provider-latentie. Dit heeft gevolgen voor de reproduceerbaarheid van cloud-metingen: de TTFT en generatietijd zijn minder stabiel dan bij lokale inferentie.

8.6. Extractiekwaliteit

8.6.1. Successrate



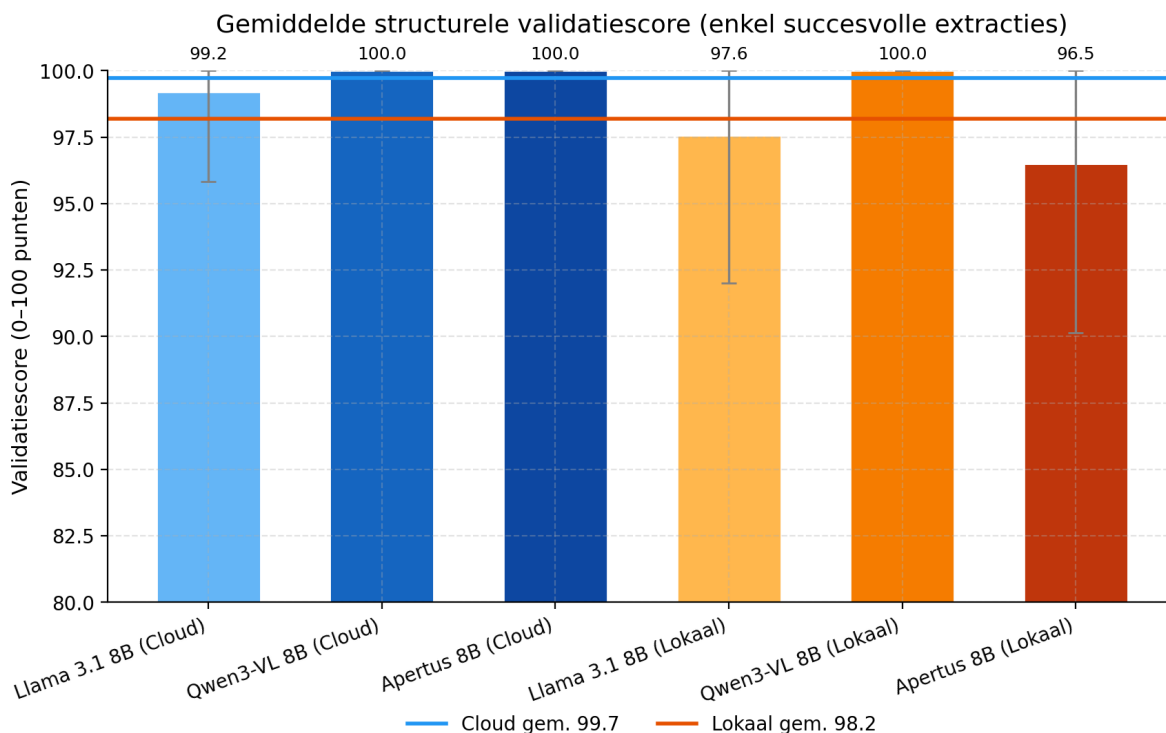
Figuur 8.16: Successrate (aandeel correct geparseerde JSON-uitvoer) per model. Gemiddeld cloud: 92,5%; gemiddeld lokaal: 89,7%. Qwen3-VL 8B lokaal en Apertus 8B cloud bereiken 100%; Apertus 8B lokaal scoort significant lager (71,4%), wat wijst op een kwantisatie-artefact. Percentage-labels boven elke balk.

Figuur 8.16 toont de successrate. **Qwen3-VL 8B lokaal** en **Apertus 8B cloud** bereiken een successrate van **100%**: elk gegenereerd antwoord kon correct worden geparseerd en voldeed aan het Pydantic-schema. **Qwen3-VL 8B cloud** scoort **~85%**, wat aangeeft dat de cloud-implementatie voor dit model de `response_format`-afdwinging minder strikt toepast dan de lokale grammaticagestuurde variant.

Llama 3.1 8B vertoont bij zowel cloud (92,4%) als lokaal (97,8%) enige parseerverlies, wat duidt op incidentele afwijkingen van de gevraagde JSON-structuur ondanks grammaticagestuurde decoding.

Lokaal scoort Apertus 8B significant slechter (71,4%) dan zijn cloud-tegenhanger (100%), wat wijst op een kwantisatie-artefact dat de structurele outputkwaliteit aantast. Dit is een belangrijk voorbeeld van hoe kwantisatie niet alleen de energie-efficiëntie beïnvloedt, maar ook de functionele correctheid van het model kan ondermijnen.

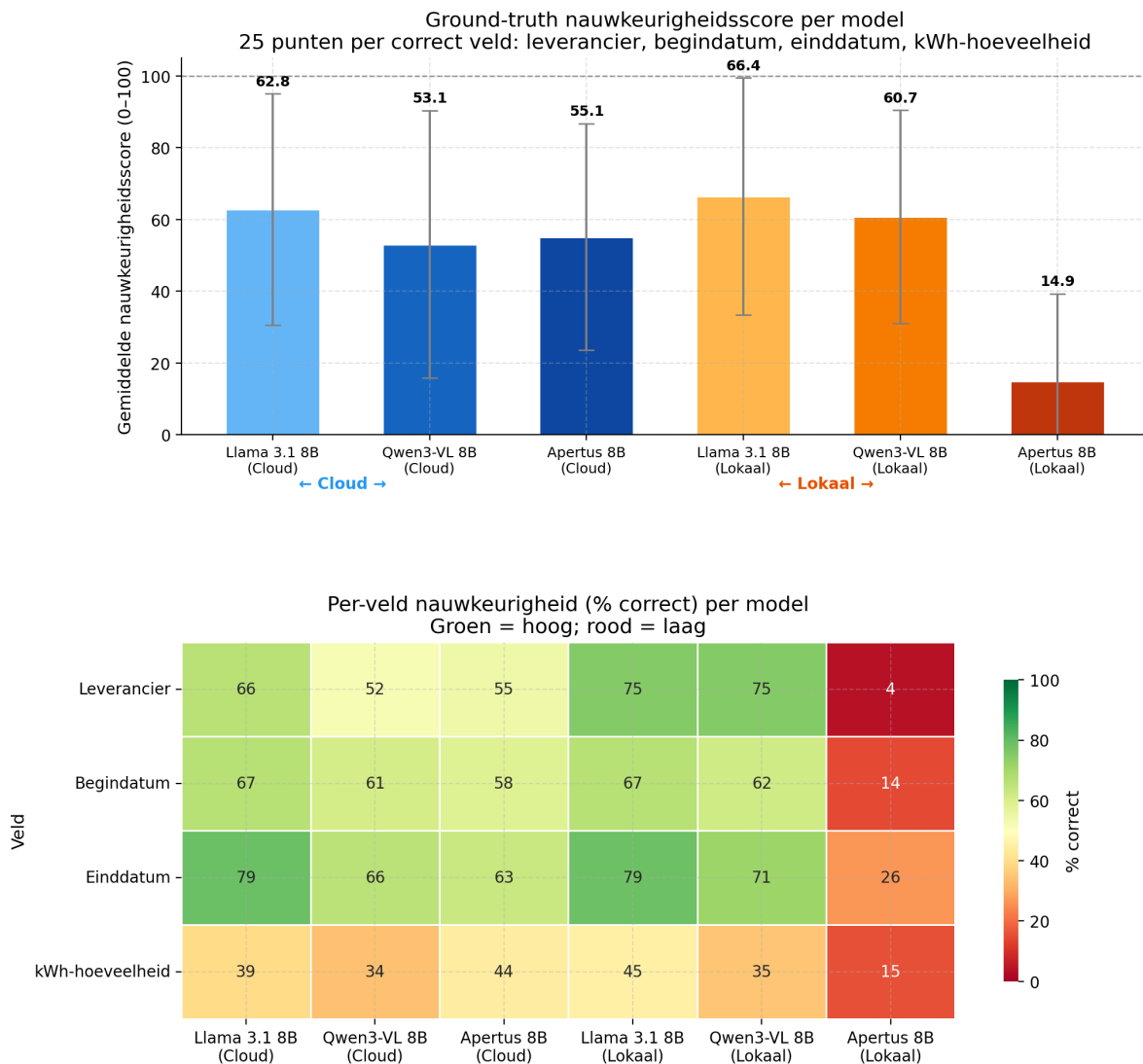
8.6.2. Structurele validatiescore



Figuur 8.17: Gemiddelde structurele validatiescore (0–100 punten) per model, berekend uitsluitend over succesvolle extracties. Qwen3-VL 8B (cloud en lokaal) en Apertus 8B cloud scoren consistent 100/100; Llama 3.1 8B scoort $\approx 97\text{--}99$; Apertus 8B lokaal scoort $\approx 96,5$, wat structurele fouten aangeeft die verband houden met het kwantisatie-artefact. Foutbalken: $\pm 1\sigma$, bovengrens begrensd op 100.

Figuur 8.17 richt zich op de kwaliteit van de succesvolle extracties. **Qwen3-VL 8B** (cloud én lokaal) en **Apertus 8B** (cloud) scoren consistent **100/100**: alle verplichte velden zijn aanwezig met het correcte datatype. Llama 3.1 8B scoort iets lager ($\approx 97\text{--}99$), wat wijst op incidentele type-afwijkingen in het `kwh_quantity`-veld (numeriek vs. string). De hoge scores bevestigen dat grammaticagestuurde decodering de structurele kwaliteit van de output effectief borgt, ongeacht of het model lokaal of via de cloud wordt aangesproken. Opnieuw valt de lokale versie van Apertus 8B op met een lagere validatiescore ($\approx 96,5$), wat wijst op structurele fouten die mogelijk verband houden met het kwantisatie-artefact dat ook de successrate aantast.

8.6.3. Inhoudsnauwkeurigheid (ground-truth vergelijking)



Figuur 8.18: Ground-truth nauwkeurigheidsscore per model (boven; 25 punten per correct veld: leverancier, begindatum, einddatum, kWh-hoeveelheid) en per-veld nauwkeurigheidshetmap (onder; groen = hoog, rood = laag). Apertus 8B lokaal is met 14,9/100 een uitschieter ondanks een hoge structurele validatiescore, wat illustreert dat structureel correct $\neq$ inhoudelijk correct. Foutbalken: $\pm 1\sigma$, bovengrens begrensd op 100.

Figuur 8.18 toont voor het eerst niet *of* de extractie structureel geldig is, maar *hoe correct* de geëxtraheerde waarden zijn ten opzichte van de bekende juiste waarden. Twee bevindingen springen in het oog.

Apertus 8B lokaal is een uitschieter. Met een gemiddelde nauwkeurigheidsscore van slechts 14,9/100 (tegenover 55–67 voor de overige modellen) presteert dit model dramatisch slechter op feitelijke extractie dan zijn cloud-tegenhanger (55,1). Het model genereert weliswaar syntactisch geldige JSON (de structurele validatiescore is hoog, zie Figuur 8.17), maar de geëxtraheerde veldwaarden zijn grotendeels incor-

rect. Dit illustreert een cruciaal verschil: **structureel correct** $\neq$ **inhoudelijk correct**. Grammaticagestuurde decodering garandeert een valide schema, maar borgt niet de inhoudelijke kwaliteit van de geëxtraheerde informatie.

kWh-hoeveelheid is het moeilijkste veld. De heatmap toont dat de nauwkeurigheid voor `kwh_quantity` (kWh-hoeveelheid) met 15–45% aanzienlijk lager ligt dan voor datumvelden (14–79%) en leveranciersnamen (4–75%). Dit is verklaarbaar: facturen voor brandstoffen en afvalverwerking bevatten meerdere kwantitatieve getallen (volumes, eenheidsprijzen, BTW-bedragen), waardoor modellen de verkeerde waarde oppikken. De goede datumnauwkeurigheid bij Llama-modellen (67–79%) is deels toe te schrijven aan de datumnormalisatie: het model schrijft datums in het DD/MM/YYYY-formaat, maar de normalisatiestap herkent dit correct als gelijkwaardig aan de ISO 8601-referentiewaarden.

Llama 3.1 8B scoort het best op inhoudsnauwkeurigheid (cloud: 62,8; lokaal: 66,4), ondanks zijn iets lagere successrate (zie Figuur 8.16). Dit suggereert dat modellen die structureel iets vaker falen, bij succesvolle aanroepen doorgaans correctere waarden extraheren dan modellen die altijd een geldig schema genereren maar met inhoudelijk zwakkere waarden.

Hypothese: impact van 4-bit kwantisatie op de modelarchitectuur van Apertus.

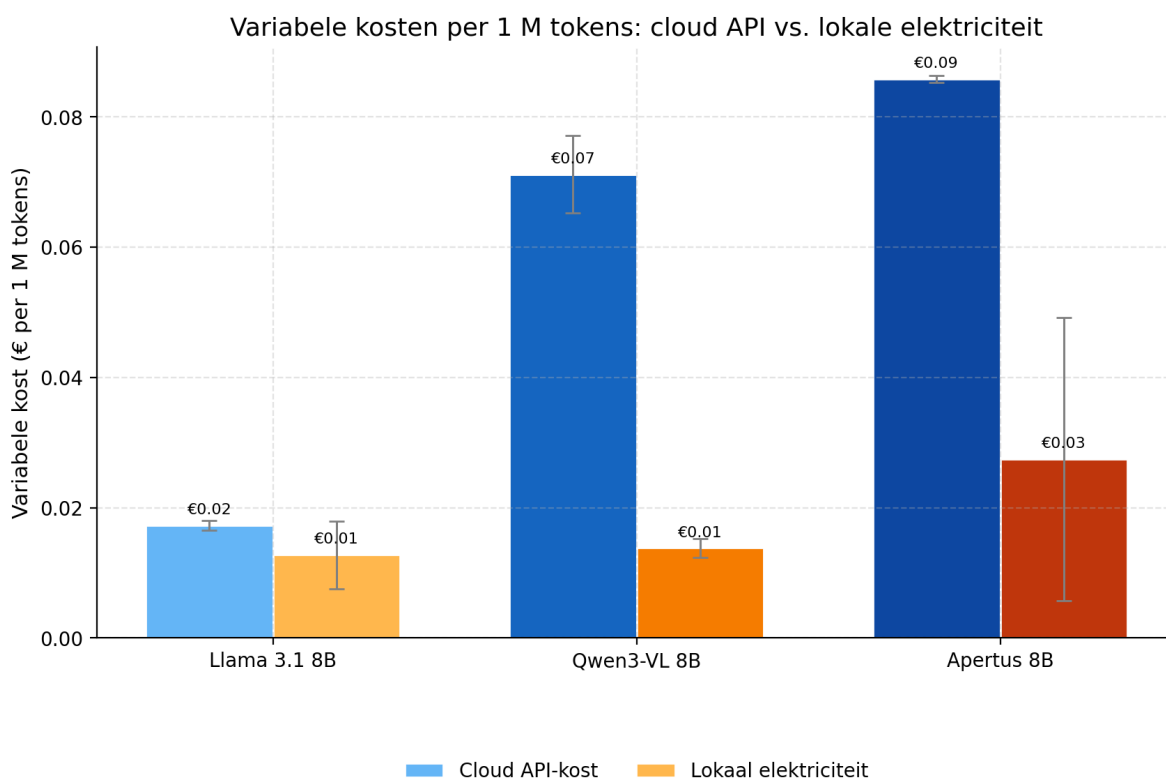
Een aanvullende verklaring voor het dramatische kwaliteitsverschil van Apertus 8B tussen cloud (FP16, score 55,1) en lokaal (Q4_K_M, score 14,9) is niet zonder meer te zoeken in de ondervertegenwoordiging van het Italiaans in het trainingskorpus. De technical report van Apertus beschrijft juist een expliciet multilinguale opzet: het model is voorgetraind op 15T tokens uit meer dan 1.800 talen, waarvan $\approx 40\%$ niet-Engelstalig materiaal, en er is post-training data opgenomen voor 149 talen. Italiaans valt volledig binnen de EU-evaluatietests waarop Apertus getoetst is. De hypothese dat Italiaans specifiek ondervertegenwoordigd is in dit trainingskorpus vindt dus geen steun in de beschikbare documentatie en zou voor Apertus minder gelden dan voor modellen die uitsluitend op Engelstalige corpora zijn getraind (Apertus e.a., 2025).

Een plausibeler verklaring voor de kwaliteitsdaling bij lokale Q4_K_M-kwantisatie ligt in de **niet-standaard architectuurelementen** van Apertus. Het model introduceert onder meer een eigen activatiefunctie (xIELU) en QK-Normalisatie, die afwijken van de standaard LLaMA-architectuur waarvoor de GGUF-kwantisatiepijplijn van `llama.cpp` primair geoptimaliseerd is (Gerganov & contributors, 2023). Bij 4-bit kwantisatie worden modelgewichten agressief gecomprimeerd; het is plausibel dat deze compressie de numerieke stabiliteit van niet-standaard activatiepatronen sterker aantast dan die van gevestigde LLaMA-compatibele architecturen (Khan e.a., 2025; Wan e.a., 2023). Daarnaast is Apertus een recent model (arXiv-preprint september 2025), en de GGUF-kwantisatiegemeenschap heeft doorgaans meer tijd nodig om nieuwe architecturen optimaal te ondersteunen dan voor modellen

als Llama 3.1 die al langer beschikbaar zijn (Apertus e.a., 2025). Dit verklaart tevens waarom de structurele kwaliteit (JSON-schema) hoog blijft terwijl de inhoudsnauwkeurigheid instort: kwantisatiefouten treffen eerder de fijne semantische extractie dan de grove structuurvorming. Beide hypothesen verdienen empirisch vervolgonderzoek, bij voorkeur via een vergelijking van kwantisatieniveaus (Q4 vs. Q8 vs. FP16) op meertalige extractietaken, en via een directe vergelijking met architectuurcompatibele kwantisatietools.

8.7. Kosten en break-even analyse

8.7.1. Variabele kosten per 1 miljoen tokens



Figuur 8.19: Gemiddelde variabele kosten per 1 M tokens: cloud API-tarief (Hugging Face, 2026) versus lokale elektriciteitskosten ($\text{energy_kwh_with_pue} \times \text{€0,2377/kWh}$ (Eurostat, 2022)), per modelkoppel. Foutbalken op cloud-kolommen weerspiegelen variatie in de input/output-tokenverhouding per aanroep; foutbalken op lokale kolommen weerspiegelen variatie in energieverbruik. Waarden per balk vermeld in €. Voor Llama 3.1 8B ligt de effectieve cloud-kost in dezelfde orde van grootte als de lokale elektriciteitskosten.

Figuur 8.19 vergelijkt de variabele kosten per 1 M tokens. Dit is de meest correcte vergelijkingsbasis: cloud-tarieven worden per definitie per 1 M tokens gepubliceerd (zie Tabel 5.1), en lokale elektriciteitskosten worden hier op dezelfde eenheid genormaliseerd als $\frac{\text{energy_kwh_with_pue} \times \text{€0,2377}}{\text{total_tokens}} \times 10^6$. Een essentieel onderscheid blijft: de cloud API-kost is een *volledig variabele* kost die per token wordt aangere-

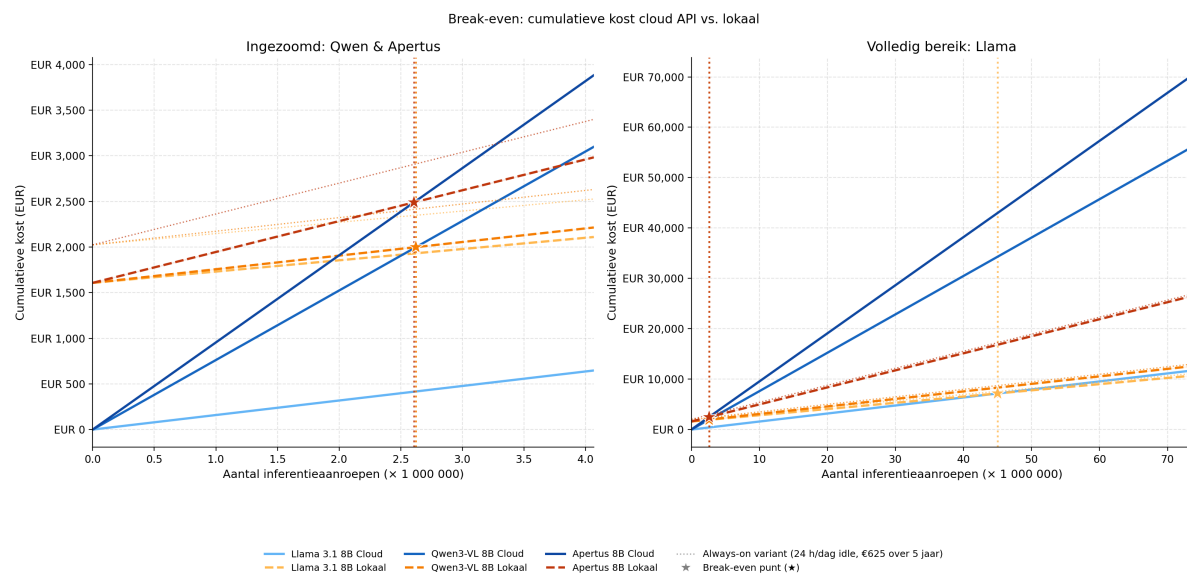
kend, terwijl de lokale elektriciteitskosten de *variabele operationele kost* zijn naast een *vaste hardware-investering* van €1.400.

De foutbalken op de cloud-kolommen weerspiegelen geen prijsonzekerheid — de API-tarieven liggen per definitie vast — maar de variatie in de verhouding inputtokens tot outputtokens per aanroep. Omdat beide tokentypen tegen een verschillend tarief worden aangerekend, varieert de effectieve kost per totaaltoken naargelang de token-mix van elke aanroep. Bij lokale kolommen zijn de foutbalken wel een directe maat voor variabiliteit in energieverbruik.

De tariefverschillen tussen cloud-modellen zijn groot: Llama 3.1 8B heeft een effectieve gemiddelde kost die sterk gedomineerd wordt door het inputtarief (€0,0170/1M tok), omdat de invoerprompts bij documentextractie gemiddeld $\approx 150\times$ langer zijn dan de output. Qwen3-VL 8B en Apertus 8B zijn aanzienlijk duurder door hogere output- en inputtarieven.

Kritische observatie: voor Llama 3.1 8B ligt de effectieve cloud API-kost per token *in dezelfde orde van grootte* als de lokale elektriciteitskosten per token, waarmee de cloud vanuit een zuiver OpEx-perspectief nauwelijks duurder is bij dit belastingsprofiel. Dit is een direct gevolg van de bijzonder scherpe inputtarieven van HuggingFace Hub voor dit model in combinatie met de lange inputprompts die de effectieve kostprijs per token naar het inputtarief trekken.

8.7.2. Break-even analyse: alle modelkoppels



Figuur 8.20: Cumulatieve kostenvergelijking cloud API versus lokaal (vaste hardware €1.400 + idle-elektriciteit €208 over 5 jaar + variabele elektriciteitskosten (Eurostat, 2022)) voor de drie modelkoppels. Links: ingezoomd bereik (Qwen & Apertus, BEP $\approx 2,6$ M aanroepen); rechts: volledig bereik (Llama, BEP ≈ 45 M aanroepen). Gestippelde lijnen: altijd-actieve variant (24 h/dag idle, €625 over 5 jaar, BEP $\approx 3,3$ M voor Qwen/Apertus). Ster (★) = break-even punt; gekleurde band: $\pm 1\sigma$ (schaalt met $\sqrt{n}$). Cloud API-tarieven: (Hugging Face, 2026).

Figuur 8.20 vergelijkt de break-even analyse voor alle drie modelkoppels. De per-aanroep variabele kosten die aan de berekening ten grondslag liggen zijn:

Model	Cloud API (m€/aanroep)	Lokaal elektriciteit (m€/aanroep)
Llama 3.1 8B	0,1592	0,1235
Qwen3-VL 8B	0,7620	0,1491
Apertus 8B	0,9549	0,3381

Tabel 8.1: Gemiddelde variabele kost per aanroep (basis voor de break-even berekening). Hardware €1.400 is vaste kost en niet inbegrepen in de bovenstaande waarden.

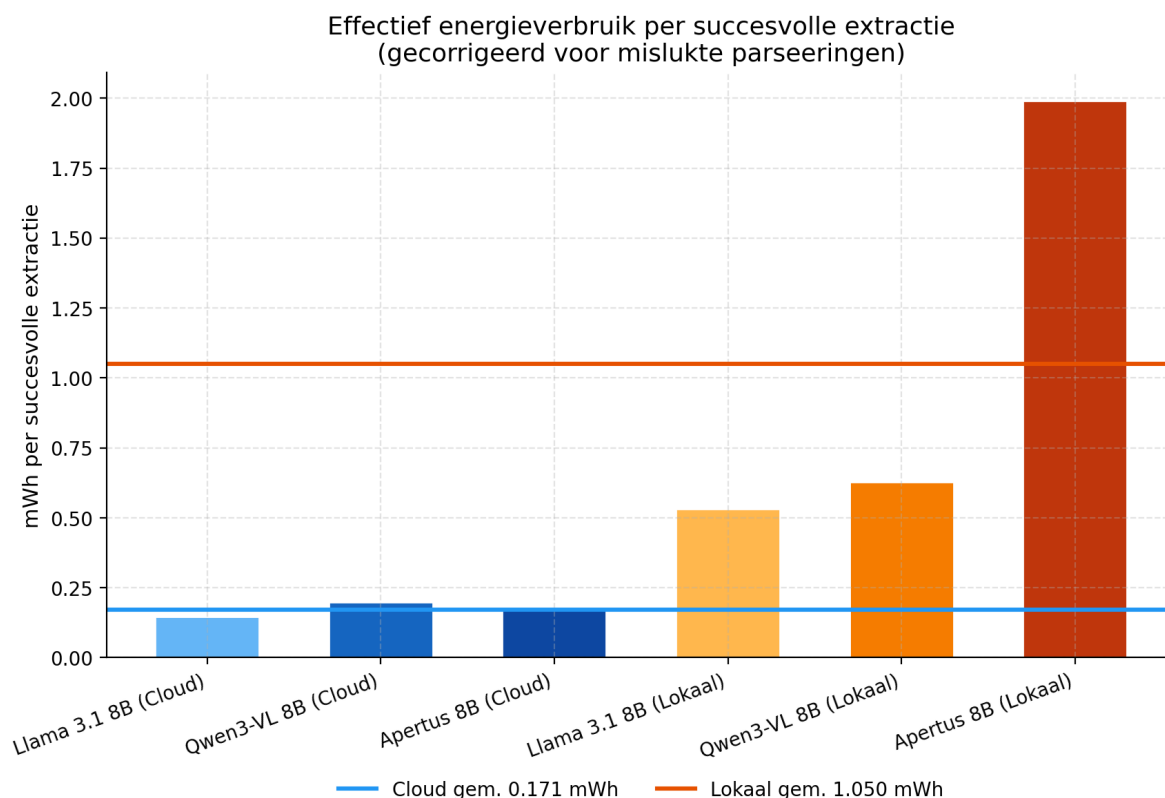
Het beeld verschilt sterk naargelang de tariefstructuur:

- **Llama 3.1 8B:** BEP $\approx$ 45 miljoen aanroepen (praktisch onhaalbaar). De cloud API-kost (0,159 m€/aanroep) ligt *hoger* dan de lokale elektriciteitskosten alleen (0,124 m€), maar het verschil is zo klein (0,036 m€/aanroep) dat de gecombineerde vaste kost van hardware (€1.400) en idle-elektriciteit (€208 over 5 jaar, zie Sectie 5.1) pas na $\approx$ 45 miljoen aanroepen terugverdiend is.
- **Qwen3-VL 8B:** BEP $\approx$ 2,62 miljoen aanroepen. Bij een volume van 1.000 aanroepen per dag bedraagt de terugverdientijd $\approx$ 7,2 jaar.
- **Apertus 8B:** BEP $\approx$ 2,61 miljoen aanroepen. Vergelijkbaar met Qwen, maar met een hogere absolute kost per aanroep.

De altijd-actieve variant (gestippelde lijnen in Figuur 8.20) vervangt de standaard idle-kost van €208 (8 u/dag) door de volledige idle-kost van €625 over 5 jaar (60 W $\times$ 24 u/dag, zie Sectie 5.1) en verschuift de BEP-punten voor Qwen en Apertus naar respectievelijk $\approx$ 3,3 miljoen aanroepen.

Implicatie voor Turtle Srl: bij een typisch verwerkingsvolume van tientallen tot honderden facturen per dag levert cloud-inferentie een lagere totale eigendoms-kost op. Lokale inferentie wordt pas financieel interessant bij hoge verwerkingsvolumes (meerdere duizenden aanroepen per dag), en dan uitsluitend voor de duurdere cloudmodellen (Qwen, Apertus).

8.7.3. Kosten per succesvolle extractie



Figuur 8.21: Energieverbruik per *succesvolle* extractie (mWh), berekend als gemiddeld energieverbruik per aanroep gedeeld door de successrate per model. Qwen3-VL 8B (cloud en lokaal) scoort het best door de combinatie van lage energie én 100 % successrate. Apertus 8B lokaal heeft de hoogste gecorrigeerde waarde door zowel hoog energieverbruik als lage successrate (71,4 %). Referentielijnen tonen het groeps gemiddelde cloud en lokaal.

Figuur 8.21 corrigeert het energieverbruik voor het faalpercentage: het toont de energie per *succesvolle* extractie. Dit is relevant voor de praktijk, omdat een mislukte extractie opnieuw verwerkt moet worden. Qwen3-VL 8B (cloud en lokaal) scoort hier het best door de combinatie van lage energie én een successrate van 100%. Llama 3.1 8B cloud krijgt door zijn lagere successrate (≈92,4%) een licht hogere gecorrigeerde waarde dan zijn ruwe energiemeting, maar blijft desondanks de zuinigste cloudoptie.

8.8. Statistische toetsen

Om de significantie van de belangrijkste vergelijkingen te onderbouwen werden Welch-t-toetsen uitgevoerd (ongelijke varianties, tweezijdig). De Welch-t-toets is ook bij grote steekproeven de standaardkeuze voor twee-steekproefvergelijkingen met onbekende populatievariantie; bij de hier aanwezige steekproefomvang ($n > 3000$ per groep) is de t-verdeling met meer dan 3000 vrijheidsgraden praktisch onderscheidbaar van de standaard normaalverdeling, zodat het onderscheid met

een z-toets geen meetbaar effect heeft op de p -waarden. Tabel 8.2 vat de resultaten samen.

Tabel 8.2: Resultaten van de Welch- t -toetsen voor de vergelijking cloud vs. lokaal.

Variabele	t -waarde	p -waarde	Conclusie
Energieverbruik per aanroep (kWh)a	-42,5	$< 10^{-300}$	Significant ($p \ll 0,001$)
Inferentietijd per aanroep (s)	-38,6	$< 10^{-300}$	Significant ($p \ll 0,001$)
CO ₂ per aanroep (kgCO ₂ eq)a	-42,5	$< 10^{-300}$	Significant ($p \ll 0,001$)

De Welch- t -toets is uitgevoerd op de absolute waarden per aanroep (energy_kwh_with_pue resp. co2_kg). Omdat alle modellen dezelfde 21 documenten verwerken, is de gemiddelde tokenlengte per groep vergelijkbaar, waardoor de t -statistiek op per-aanroepwaarden nagenoeg identiek is aan die op genormaliseerde per-token-waarden.

De uiterst lage p -waarden ($\ll 0,001$) voor alle drie de kernvariabelen bevestigen dat de gemeten verschillen tussen cloud- en lokale inferentie statistisch significant zijn en niet op toeval berusten. De grote absolute t -waarden ($|t| > 42$) zijn te verklaren door de grote steekproefomvang ($n > 3000$ per groep) en de consistente richting van het verschil.

8.9. Synthese van de resultaten

De experimentele resultaten convergeren naar een coherent beeld dat de hypothesen uit de literatuurstudie (Hoofdstuk 2) zowel bevestigt als nuanceert:

Energie en CO₂

Cloud-inferentie is per 1000 tokens energiezuiniger dan lokale inferentie voor alle geteste modellen (verhouding $\approx 3,7$ tot $\approx 8,5$, afhankelijk van het model; gemiddeld $\approx 5\times$, zie Figuur 8.1); ten opzichte van de cloud-ondergrens ($\alpha = 0$) loopt dit verschil op tot $>100\times$. De werkelijke cloud-energie bevindt zich in een bandbreedte tussen beide waarden. Dit is consistent met de literatuur (Fernandez e.a., 2025; Samsi e.a., 2023): enterprise-hardware en serverside batching resulteren in een hogere energie-efficiëntie per token dan consumentenhardware. De Caravaca-prefill-correctie ($\kappa_{\text{gem}} \approx 30,7$) blijkt cruciaal om een vertekend beeld te vermijden, en is specifiek voor het belastingsprofiel van documentextractie (lange input, korte output). Een methodologische kanttekening bij de lokale energiewaarden: deze zijn gebaseerd op CodeCarbon-metingen en zijn onderhevig aan de inherente meetafwijking van deze bibliotheek. Externe validatiestudies plaatsen de afwijking op tot circa 40% ten opzichte van directe hardwaremetingen (Fischer, 2025). De lokale balken in de grafieken representeren bijgevolg geschatte waarden, niet exacte metingen, analoog aan de bandbreedte die voor cloud-schattingen wordt gevisuali-

seerd. De richting van het verschil (lokaal > cloud) en de relatieve rangschikking tussen modellen zijn robuust voor deze meetafwijking, omdat alle lokale modellen via dezelfde methode zijn gemeten.

Kwaliteit

De structurele extractiekwaliteit is voor de meeste modellen zeer hoog ($\geq 97\%$) en vertoont geen systematisch verschil tussen cloud en lokaal. Grammaragestuurde decoding borgt de structurele validiteit effectief in beide infrastructuurscenario's; de gemeten scores van $\geq 97\%$ tonen aan dat grammatica-gebonden decoding in dit toepassingsscenario geen kwaliteitsverlies oplevert. De inhoudsnauwkeurigheid (ground-truth vergelijking, Figuur 8.18) toont echter een belangrijke nuance: structurele geldigheid impliceert *niet* inhoudelijke correctheid. Apertus 8B lokaal scoort 14,9/100 op de ground-truth vergelijking ondanks een hoge structurele score, terwijl Llama 3.1 8B — met de hoogste inhoudsnauwkeurigheid (63–66) — iets vaker structureel faalt. kWh-hoeveelheid is het moeilijkste veld (15–45% correct), wat het belang onderstreept van een expliciete ground-truth evaluatie naast structurele validatie.

Latentie

Cloud-inferentie is gemiddeld $\approx 2,3\times$ sneller dan lokale inferentie bij dit belastingsprofiel. De doorvoeranalyse (Figuur 8.14) maakt duidelijk dat dit voordeel vrijwel volledig in de **prefill-fase** zit: cloud verwerkt inputtokens $\approx 2,2\times$ sneller (≈ 5.835 vs. 2.678 tok/s). De **decode-doorvoer** is daarentegen bijna gelijk (≈ 51 vs. 39 tok/s, factor $\approx 1,3$), omdat tokengewijze generatie inherent serieel is. Apertus 8B lokaal is een uitzondering met bijzonder trage inferentie ($\approx 16,7$ s gemiddeld).

Kosten

De break-even analyse levert het meest verrassende resultaat: voor het goedkoopste cloud-model (Llama 3.1 8B) is de cloud-API zelfs per-aanroep goedkoper dan de lokale elektriciteitskosten, waardoor lokale hardware voor dit koppel financieel nooit terugverdiend wordt bij realistisch KMO-volume. Voor Qwen3-VL 8B en Apertus 8B, waarvan de cloud-tarieven aanzienlijk hoger liggen, is het break-even punt bereikbaar bij $\approx 2,6$ miljoen aanroepen (inclusief idle-kost van €208 over 5 jaar), wat bij een dagelijks volume van 1.000 aanroepen neerkomt op een terugverdientijd van ≈ 7 jaar.

Locatie-afhankelijkheid

De CO₂-balans is sterk locatiegebonden. Voor een lokale opstelling in Italië ($0,285$ kgCO₂eq/kWh) is een Frans of Belgisch datacenter ecologisch gunstiger dan het lokale alternatief. Dit onderstreept de aanbeveling uit Sectie 2.4: de datacenterlocatie van de cloudprovider is een relevante variabele bij de architectuurkeuze.

9

Beslissingskader voor duurzame AI-integratie

De experimentele bevindingen uit Hoofdstuk 8 leveren een multidimensionale dataset op die verder gaat dan de loutere vaststelling dat cloud-inferentie energiezuiniger is dan lokale inferentie. De keuze tussen een cloud-API en een lokaal gehost model kent immers geen universeel antwoord: ze is afhankelijk van de technische, economische en organisatorische context van elke individuele deployment. Dit hoofdstuk vertaalt de empirische resultaten naar een praktisch en direct inzetbaar beslissingskader voor KMO's, IT-architecten en webontwikkelaars die LLM-functionaliteit wensen te integreren in een webapplicatie. Het centrale instrument is een beslissingsboom die de architectuurkeuze stapsgewijs stuurt op basis van negen meetbare contextparameters. Daarmee wordt tevens een antwoord geformuleerd op de deelvraag of de keuze toepassingsscenario-afhankelijk is, dan wel of er sprake is van een structureel voordeel voor één van beide opties.

9.1. Van empirische data naar beslissingsvariabelen

Uit de analyse in Hoofdstuk 8 kunnen negen variabelen worden geïdentificeerd die de voorkeur voor cloud of lokaal significant beïnvloeden. Tabel 9.1 geeft een overzicht van deze variabelen, hun empirische grondslag en de richting van hun effect op de architectuurkeuze.

Variabele	Empirische grondslag	Effect op keuze
Dataprivacy	Functionele requirement Turtle Srl (Sectie 5.1)	Mandateert lokaal indien geen anonimisering haalbaar is
GPU-beschikbaarheid	RTX 3070, 8 GB VRAM (Tabel 4.1)	Ontbrekende hardware sluit het lokale pad technisch uit
Dagelijks volume	Break-even analyse (Figuur 8.20)	Hoog volume maakt lokale hardware financieel interessant
Inputlengte (tokens)	Scatter- en prefill-analyse (Figuren 8.4 en 8.5)	Lange input verhoogt absolute cloud-kosten per aanroep via κ
Cloud-API-tarief	Tabel 5.1 en 8.1	Hoge tarieven verlagen de BEP-drempel
BEP-haalbaarheid	Break-even analyse (Figuur 8.20)	BEP bereikbaar binnen 7 jaar bij huidig volume $\Rightarrow$ lokale hosting economisch rendabel
Real-time latentie	Inferentietijdanalyse (Figuur 8.12)	Interactieve apps vereisen cloud (2,3 $\times$ sneller)
Lokale stroommix	Regionale gevoeligheidsanalyse (Figuur 8.10)	Groene lokale stroom neutraliseert het cloud-CO ₂ -voordeel
Kwantisatiekwaliteit	Ground-truth analyse (Figuur 8.18)	4-bit kwantisatie kan inhoudskwaliteit degraderen bij meertalige taken

Tabel 9.1: Negen beslissingsvariabelen voor de keuze tussen lokale en cloud-LLM-inferentie, met empirische grondslag en richtingseffect.

9.2. Belastingsprofiel als proxy voor het toepassingsscenario

De tweede deelvraag in het probleemdomein luidt: *hangt de keuze af van het toepassingsscenario binnen de applicatie, of is de ene optie beter in alle scenario's?* Dit onderzoek evalueerde uitsluitend het toepassingsscenario *Document Information Extraction*, waardoor een directe empirische vergelijking over meerdere applicatiedomeinen niet mogelijk is. Toch laten de resultaten toe om een werkzaam antwoord te formuleren via het concept **belastingsprofiel**: een set kwantificeerbare kenmerken die onafhankelijk zijn van de specifieke domeincontext en toch het energetische en economische gedrag van elke infrastructuuroptie volledig beschrijven.

Een belastingsprofiel wordt bepaald door drie meetbare dimensies:

1. Verhouding inputtokens tot outputtokens ($r = N_{in}/N_{out}$)

Dit is de meest bepalende karakteristiek voor de energievergelijking. Bij het documentextractie-toepassingsscenario van dit onderzoek bedraagt $r \approx 150$, wat resulteert in een gemiddelde prefill-correctiefactor $\kappa_{gem} \approx 30,7$ (Sectie 8.2). Bij toepassingen waarbij input en output in dezelfde orde van grootte liggen ($r \approx 1$, bijv. chatbots), nadert $\kappa \rightarrow 1$ en is de cloud-energie haar minimale waarde. De relatieve cloud-efficiëntie ten opzichte van lokaal (3,7–8,5× per token) blijft structureel gelden voor elke r , maar de *absolute* cloud-kost per aanroep schaalt mee: bij hoge r worden de cloud-aanroepen duurder, waardoor de economische vergelijking verschuift ten gunste van lokale hosting.

2. Dagelijks aanroepvolume

De break-even analyse (Sectie 8.7) toont dat het volume de terugverdientijd van de hardware bepaalt, ongeacht de applicatiedomeinstelling. Een documentextractie-toepassingsscenario met 50 aanroepen per dag scoort financieel wezenlijk anders dan een chatbot-toepassingsscenario met 50.000 aanroepen per dag, ook bij identiek tokenverbruik per aanroep.

3. Latentievereiste (real-time versus batch)

Interactieve toepassingen vereisen een perceptueel aanvaardbare responstijd; batch-georiënteerde workflows zijn hier niet aan gebonden. Uit Figuur 8.12 blijkt dat lokale inferentie gemiddeld 9,87 seconden vereist, tegenover 4,22 seconden in de cloud. Bij interactieve toepassingen met complexe documenten kan de lokale TTFT (3,4–4,5 s) al de praktische bruikbaarheidsdrempel naderen.

Door toepassingsscenario's te karakteriseren via hun belastingprofiel (r , volume, latentie) in plaats van via hun domeinnaam, kan het beslissingskader breder worden toegepast dan de specifieke context van dit onderzoek. Tabel 9.2 illustreert hoe een aantal typische AI-toepassingen worden gepositioneerd in dit driedimensionale ruimte.

Toepassingsscenario	$r = N_{in} / N_{out}$	Typisch volume	Latentie	Aanbeveling kader
Chatbot / conversatie	$\approx 1-5$	Variabel (laag-hoog)	Real-time	Cloud (lage r , hoge κ -winst relatief klein)
Documentextractie (dit onderzoek)	$\approx 100-200$	Laag-middel (batch)	Batch	Afhankelijk van volume en tarief
RAG / Q&A met context	$\approx 10-50$	Middel	Real-time	Cloud (latentiegate)
Samenvatting	$\approx 20-100$	Laag-middel (batch)	Batch	Volume- en tariefafhankelijk
Codegeneratie	$\approx 2-20$	Laag-middel	Real-time	Cloud (latentiegate)

Tabel 9.2: Typische AI-toepassingsscenario's gepositioneerd op de drie belastingsdimensies op basis van een eigen synthese. De r -waarden en volumeklassen zijn kwalitatieve schattingen; aanbevelingen zijn indicatief en hangen af van de exacte waarden per knoop van het beslissingskader.

Het antwoord op de deelvraag luidt bijgevolg: **de keuze is niet afhankelijk van het toepassingsscenario als domein, maar van de werklasterkenmerken die inherent zijn aan elk toepassingsscenario**. Cloud is in alle geteste configuraties consistent energiezuiniger per token en is bij lage volumes ook economisch de meest voor de hand liggende keuze. Lokale hosting wordt competitief naarmate het dagelijks volume stijgt, de cloud-tarieven hoog zijn, privacyvereisten aanwezig zijn, of de lokale stroom hernieuwbaar is. Er is dus geen optie die *altijd* beter is: de grenzen worden bepaald door de drie belastingsdimensies, niet door de applicatiedomeinstelling op zichzelf.

9.3. Het beslissingskader

Figuur 9.1 toont de beslissingsboom voor de keuze tussen lokale en cloudgebaseerde LLM-inferentie. De boom doorloopt negen beslissingsknopen (G1-G9), geordend van harde constraints — die de architectuur onmiddellijk bepalen — naar economische, ecologische en kwaliteitsoverwegingen. Elke eindknoop duidt op een aanbevolen architectuurtype: **CLOUD** (blauw), **LOKAAL** (groen), **HYBRIDE** (oranje) of **VALIDEER EERST** (geel-oranje). Alle drempelwaarden zijn empirisch verankerd in de resultaten van dit onderzoek en worden in Sectie 9.4 kwantitatief onderbouwd.



Figuur 9.1: Beslissingsboom voor de keuze tussen lokale en cloudgebaseerde LLM-inferentie in een webapplicatie. Groene eindknoten: lokale hosting aanbevolen; blauwe eindknoten: cloud-API aanbevolen; oranje eindknoop: hybride architectuur; geel-oranje eindknoten: kwaliteitsvalidatie vereist vóór uitrol.

9.4. Bespreking van de beslissingsknopen

Elke beslissingsknoop in de boom is empirisch verankerd in de resultaten van Hoofdstuk 8. Onderstaande bespreking legt de kwantitatieve grondslag voor elke drempelwaarde en licht toe waarom de knopen in deze volgorde zijn geplaatst.

9.4.1. Knoop G1 – Dataprivacy

De eerste beslissingsknoop is een harde *binary gate*: bevatten de te verwerken documenten privacygevoelige of vertrouwelijke informatie? In de casus van Turtle Srl gaat het om bedrijfsgevoelige energiefacturen en duurzaamheidsrapporten met klant- en financiële data. Indien een anonimiseringspreprocessing haalbaar is (waarbij namen, adressen en unieke identificatoren worden vervangen door pseudoniemen vóór de clouddoorvoer), is een hybride architectuur mogelijk (zie Sectie 5.1). De anonimiseringsstap valt buiten de scope van dit onderzoek, maar is een relevante optie voor productieomgevingen. Zoniet is lokale hosting een harde vereiste, ongeacht de ecologische of economische vergelijking.

9.4.2. Knoop G2 – GPU-hardwarebeschikbaarheid

Lokale inferentie vereist minimaal een GPU met voldoende VRAM om het model volledig in geheugen te laden. Op basis van de configuratie in dit onderzoek is een kaart met 8 GB VRAM het minimum voor 7B–9B-klasse modellen bij Q4_K_M-kwantisatie (zie Tabel 4.1 en Sectie 4.1.3). Grotere modellen vereisen proportioneel meer geheugen. Ontbreekt die hardware, of overschrijdt de aankoopkost het beschikbare budget, dan vervalt het lokale pad en is cloud de enige realistisch inzetbare optie.

9.4.3. Knoop G3 – Dagelijks aanroepvolume

Uit de break-even analyse (Figuur 8.20) blijkt dat bij een laag volume van minder dan 100 aanroepen per dag de gecumuleerde cloud-API-kosten over een periode van 5 jaar ruim onder de totale eigendoms kost van de lokale infrastructuur blijven voor alle geteste modelkoppels. De drempel van 100 aanroepen per dag is als onderkant gekozen van het grensgebied waarbij de break-even berekening te verschuiven begint voor de duurste cloudmodellen. Beneden deze drempel is cloud vrijwel altijd de kostenefficiëntste keuze, ongeacht de verdere werklastkenmerken.

9.4.4. Knoop G4 – Gemiddelde inputlengte per aanroep

Bij korte inputprompts (minder dan 500 tokens, typisch voor chatbots of keyword-extractie) is de prefill-correctiefactor $\kappa \approx 1,09$ (bij $r = 0,5$ en $\alpha = 0,175$), wat betekent dat de onbewerkte EcoLogits-schatting al een goede benadering is van het werkelijke cloud-energieverbruik. In dit regime is de cloud-energie-efficiëntie het grootst relatief aan lokaal (3,7–8,5×) en zijn de absolute cloud-kosten per aanroep laag. Bij

lange inputdocumenten (meer dan 5.000 tokens, typisch voor documentextractie en samenvatting) stijgt κ naar waarden van 10 tot 250 (zie Figuur 8.5), waardoor de werkelijke cloud-energie per aanroep substantieel hoger uitvalt en de economische vergelijking met lokale hosting verschuift.

De lineaire relatie tussen inputlengte en energieverbruik bij lokale inferentie (Figuur 8.4) bevestigt dat ook het lokale energieverbruik stijgt met documentlengte, maar de relatieve cloud-efficiëntie per token blijft constant. Het is de absolute kostprijs per aanroep die meer samenloopt bij lange inputs, en die de economische drempel voor de volgende knopen bepaalt.

9.4.5. Knoop G5 – Cloud-API-tarief

Niet alle cloud-modellen hebben hetzelfde tariefprofiel. Zoals de kostenvergelijking in Sectie 8.7 aantoont, bedraagt de effectieve variabele kost voor Llama 3.1 8B 0,159 m€/aanroep, wat slechts marginaal hoger is dan de puur variabele lokale elektriciteitskosten van 0,124 m€/aanroep (zie Tabel 8.1). Het kostenverschil bedraagt in dit geval maar 0,036 m€/aanroep, waardoor de gecombineerde vaste kost van hardware (€1.400) en idle-elektriciteit (€208 over 5 jaar) pas na ≈ 45 miljoen aanroepen is terugverdiend, een praktisch onhaalbare drempel. Voor modellen met hogere cloud-tarieven (Qwen3-VL 8B: 0,762 m€/aanroep, Apertus 8B: 0,955 m€/aanroep) daalt de BEP naar $\approx 2,6$ miljoen aanroepen, wat bij hoge volumes bereikbaar is.

9.4.6. Knoop G6 – Real-time latentievereiste

Interactieve webapplicaties waarbij een eindgebruiker synchroon op het antwoord wacht, stellen latentie-eisen die de lokale opstelling in dit onderzoek niet steeds haalt. De gemiddelde inferentietijd voor lokale modellen bedraagt 9,87 s (met een uitschieter voor Apertus 8B van $\approx 16,7$ s), tegenover 4,22 s voor cloud (Figuur 8.12). De drempel van 5 seconden is gebaseerd op de gangbare praktijkrichtlijn voor aanvaardbare responstijd bij synchrone webinteracties; onder die grens behouden eindgebruikers hun aandacht. Bij niet-interactieve batchverwerking (zoals de nachtelijke verwerking van facturen bij Turtle Srl) is deze gate niet van toepassing en hoeft de latentie de keuze niet te bepalen.

9.4.7. Knoop G7 – Break-even bereikbaarheid

Bij hoge volumes (meer dan 1.000 aanroepen per dag) en dure cloud-modellen (Qwen/Apertus-klasse) wordt de BEP bereikt na $\approx 2,6$ miljoen aanroepen. Bij 1.000 aanroepen per dag bedraagt dit $\approx 7,2$ jaar (standaard scenario met 8 u/dag aan; zie Figuur 8.20). Of dit economisch aantrekkelijk is, hangt af van de beoogde hardware-levensduur en de verwachte prijsstabiliteit van de cloud-API. Bij 2.000 aanroepen per dag daalt de terugverdientijd naar $\approx 3,6$ jaar, een realistisch scenario voor middelgrote verwerkingspijplijnen. De altijd-aan-variant (gestippelde lijn in Figuur 8.20), waarbij de machine 24 u/dag actief is, verschuift de BEP door de hogere idle-kost

(€625 vs. €208 over 5 jaar) naar $\approx 3,3$ miljoen aanroepen.

9.4.8. Knoop G8 – Lokale stroommix

De regionale gevoeligheidsanalyse (Figuur 8.10) toont dat cloud-inferentie in alle zes geanalyseerde regio's (inclusief de koolstofintensievere regio's zoals de VS en het wereldgemiddelde) een lagere CO₂-uitstoot per token heeft dan de lokale opstelling in Italië. Dit is het gecombineerde effect van de hogere hardware-efficiëntie van datacenter-GPU's (H100/A100 tegenover RTX 3070) en een gunstigere datacenter-PUE. De uitzondering vormt een lokale opstelling die draait op hernieuwbare energie ($\geq 50\%$ groen): in dat geval nadert de effectieve lokale emissiefactor de nul, waardoor het structurele CO₂-voordeel van de cloud verdwijnt. Bedrijven met een eigen zonne-energieinstallatie of een gecertificeerd groenstroomcontract kunnen dit als differentiërende factor meenemen in hun architectuurkeuze. Beide lokale paden (G7-ja en G8-ja) worden vervolgens onderworpen aan een kwaliteitsvalidatieknoop G9.

9.4.9. Knoop G9 – Kwaliteitsvalidatie van het lokale model

Voordat een lokale deployment wordt aanbevolen, dient de extractiekwaliteit van het gekwantiseerde model te worden gevalideerd op een representatieve steekproef van de eigen data. Deze gate is ingevoerd op basis van het meest opvallende kwaliteitsresultaat van dit onderzoek: Apertus 8B in Q4_K_M-kwantisatie scoort slechts 14,9/100 op de ground-truth vergelijking voor Italiaanse brondocumenten, tegenover 55,1/100 voor de FP16-cloudvariant van hetzelfde model (Figuur 8.18). Het model genereert weliswaar syntactisch valide JSON (structurele validatiescore $\approx 96,5$), maar de feitelijk geëxtraheerde veldwaarden zijn grotendeels incorrect. Dit illustreert het fundamentele onderscheid: *structureel correct* $\neq$ *inhoudelijk correct*. De meest plausibele verklaring voor dit degradatiepatroon, uitgewerkt in Sectie 8.6.3, is niet dat 4-bit kwantisatie niet-Engelstalige taalrepresentaties onevenredig aantast; die hypothese vindt geen steun in de trainingskorpusdocumentatie van Apertus. Plausibeler is dat de niet-standaard architectuurelementen van Apertus (xIELU-activatiefunctie, QK-Norm) minder goed worden ondersteund door de GGUF-kwantisatiepijplijn van `llama.cpp`, die primair voor LLaMA-compatibele architecturen is geoptimaliseerd. Aanbevelingen die lokale hosting indiceren, zijn bijgevolg pas geldig indien het specifieke model op de specifieke taal en het specifieke documenttype is gevalideerd. Llama 3.1 8B en Qwen3-VL 8B tonen in dit onderzoek geen vergelijkbaar kwaliteitsverlies bij kwantisatie, maar dit is modelspecifiek en mag niet worden gegeneraliseerd zonder empirische verificatie op de eigen data.

Indien de kwaliteitsvalidatie niet is uitgevoerd of onvoldoende scoort, geeft de beslissingsboom de aanbeveling **VALIDEER EERST**: test het lokale model op een steekproef van ten minste 20–30 representatieve documenten vóór productie-

uitrol, en overweeg cloud als vangnet totdat de kwaliteitsdrempel is bereikt.

9.5. Toepassing op de casus Turtle Srl

Het beslissingskader wordt hieronder stapsgewijs doorlopen voor de concrete situatie van Turtle Srl, waarbij de empirische parameters uit dit onderzoek als invoer dienen.

G1 (Privacy): Turtle Srl verwerkt bedrijfsgevoelige duurzaamheidsrapporten en energiefacturen. Een anonimiseringspreprocessing is technisch haalbaar (zie Sectie 5.1), waardoor de boom niet onmiddellijk naar “LOKAAL verplicht” stuurt maar verdergaat naar G2. Wordt de preprocessing niet geïmplementeerd, dan mandateert G1 lokale hosting.

G2 (Hardware): De RTX 3070-werkstation is beschikbaar bij Turtle Srl (Tabel 4.1).

G3 (Volume): Het verwachte dagelijks volume bedraagt tientallen tot honderden facturen; een conservatieve schatting van 200 aanroepen per dag valt in het middenbereik (100–1.000/dag).

G4 (Inputlengte): De testdataset bevat documenten met gemiddeld ≈ 10.000 inputtokens na Markdown-preprocessing, ruim boven de 5.000-tokendrempel.

G6 (Latentie): De verwerking van duurzaamheidsrapporten en facturen is een batchwerklust zonder synchrone eindgebruikersinteractie. Latentie is geen harde eis; G6 stuurt niet naar cloud.

G8 (Stroommix): Turtle Srl is gevestigd in Italië, met een gemiddelde emissiefactor van $0,285 \text{ kgCO}_2\text{eq/kWh}$ in 2025. Een specifiek groenstroomcontract is niet bevestigd, zodat de standaardgridconditie van toepassing is.

Uitkomst: Het beslissingskader wijst voor de huidige situatie van Turtle Srl (200 aanroepen/dag, batch, standaard Italiaans grid, anonimisering haalbaar) op **cloud-inferentie als aanbevolen architectuur**. Cloud scoort in alle zes geanalyseerde regio's beter dan de lokale opstelling in Italië (Figuur 8.10), waarbij een Frans of Belgisch datacenter een factor 5 tot 19 gunstiger uitvalt. Economisch gezien hangt de uitkomst af van het gekozen model: voor Llama 3.1 8B is de cloud ook op **OpEx**-vlak nagenoeg gelijkwaardig aan lokale elektriciteitskosten, en is de hardware nooit terugverdiend bij dit volume. Voor duurdere modellen (Qwen, Apertus) zou lokale hosting rendabel worden bij volumes boven ≈ 1.000 aanroepen/dag met een terugverdientijd van $\approx 7,2$ jaar.

De privacy-overweging vormt hierbij de bepalende nuance: indien Turtle Srl geen anonimiseringspreprocessing implementeert, mandateert G1 alsnog een lokale of hybride architectuur, los van de ecologische en economische vergelijking.

9.6. Generaliseerbaarheid en beperkingen van het kader

Het gepresenteerde beslissingskader is empirisch gefundeerd op één specifiek toepassingsscenario (documentextractie), één hardwareconfiguratie (RTX 3070, 8 GB VRAM), en drie modelkoppels in de 7B–9B parameterklasse. Bij het toepassen van het kader buiten deze context zijn de volgende beperkingen relevant.

- **Parameterklasse:** Grotere modellen (13B, 70B, 405B) vereisen meerdere GPU's of high-end hardware en hebben een aanzienlijk ander energie- en kostenprofiel. De BEP-drempels en energieverhoudingen kunnen sterk afwijken van de hier gerapporteerde waarden.
- **Hardware-evolutie:** De RTX 3070 is een Ampere-generatie kaart uit 2020. Nieuwere GPU-generaties (Ada Lovelace, Blackwell) bieden substantieel betere energie-efficiëntie per token, waardoor het lokale pad in toekomstige hardware-contexten gunstiger kan uitvallen dan hier gemeten. De verhoudingen in de beslissingsboom zijn dan ook gevoelig voor de gehanteerde hardwarespecificaties.
- **Cloud-tariefevolutie:** API-prijzen dalen historisch gezien snel naarmate de modelinfrastructuur volwassener wordt. De BEP-berekeningen zijn gebaseerd op tarieven per 8 mei 2026 (zie Tabel 5.1) en kunnen snel verouderen. Een gevoeligheidsanalyse van de BEP ten opzichte van tariefveranderingen is aan te bevelen bij elke herevaluatie van het kader.
- **Lokale stroommixevolutie:** Italië heeft een groeiend aandeel hernieuwbare energie. Indien de lokale emissiefactor structureel daalt, verschuift Knoop G8 ten gunste van lokaal, ook zonder dat er expliciet een groenstroomcontract is afgesloten.
- **Kwantisatiekwaliteit voor meertalige taken:** De ground-truth analyse (Sectie 8.6.3) toont aan dat Apertus 8B bij Q4_K_M-kwantisatie dramatisch slechter scoort op feitelijke extractie uit Italiaanse documenten (14,9/100 lokaal vs. 55,1 cloud). Knoop G9 vangt dit risico op door kwaliteitsvalidatie als expliciete voorwaarde te stellen vóór een lokale uitrol, maar de drempelwaarde voor “aanvaardbare kwaliteit” is toepassingsafhankelijk en dient door de gebruiker zelf te worden gedefinieerd op basis van de vereiste accuratesse voor de specifieke bedrijfstoepassing.

Ondanks deze beperkingen biedt het kader een robuust vertrekpunt: de structuur van de beslissingsboom (privacy → hardware → volume → tokenlengte → tarief → latentie → stroommix) is architectuuronafhankelijk en kan worden bijgesteld zodra nieuwe empirische data beschikbaar komen over andere modellen, hardware-generaties of elektriciteitsmixen.

10

Conclusie

Deze bachelorproef onderzocht de ecologische voetafdruk, kostenefficiëntie en extractiekwaliteit van lokale versus cloudgebaseerde LLM-inferentie voor het toepassingsscenario *Document Information Extraction* in een webapplicatie, met de casus van Turtle Srl als praktische anker. Over 6241 inferentieaanroepen op Italiaanse energiefacturen werd aangetoond dat cloud-inferentie via de HuggingFace Hub per 1000 tokens 3,7 tot 8,5 keer energie-efficiënter is (gemiddeld $\approx 5\times$) dan lokale inferentie op een NVIDIA RTX 3070-werkstation, en dat de CO₂-uitstoot in alle geteste datacenterregio's lager uitvalt dan de lokale waarde. De cloud is bovendien gemiddeld 2,3× sneller per aanroep en levert voor de meeste modellen een vergelijkbare of hogere structurele extractiekwaliteit. Kosten vormen echter een genuanceerder verhaal: voor het goedkoopste cloud-model (Llama 3.1 8B) is lokale hardware financieel praktisch nooit rendabel bij realistisch KMO-volume, terwijl voor duurdere cloud-modellen een break-even bereikbaar is bij $\approx 2,6$ miljoen aanroepen. De locatie van het datacenter en de privacyvereisten van een bedrijf zijn minstens even bepalend als de loutere energiemeting. De bevindingen zijn samengevat in een beslissingskader dat IT-managers stapsgewijs begeleidt bij hun infrastructuurkeuze op basis van negen contextuele parameters. Tegelijkertijd heeft dit onderzoek nieuwe vragen blootgelegd die uitnodigen tot verder werk, waaronder de impact van kwantisatie op meertalige extractie, de validatie van softwaregebaseerde energiemetingen via directe hardwaremetingen, en de verfijning van de prefill-correctiefactor voor cloud-energieschattingen.

10.1. Antwoord op de onderzoeksvragen

Hoe beïnvloedt de integratie van LLM's in webontwikkelingsframeworks de ecologische voetafdruk in vergelijking met het gebruik van gecentraliseerde generatieve AI-diensten?

De resultaten van dit onderzoek tonen aan dat cloudgebaseerde LLM-diensten, bij het toepassingsscenario van *Document Information Extraction* op energie-intensieve documenten, gemiddeld **3,7 tot 8,5 keer energie-efficiënter** zijn dan lokale inferentie op een NVIDIA RTX 3070-werkstation. Deze verhouding geldt ten opzichte van de gecorrigeerde bovengrens van de cloud-energieschatting ($\alpha = 0,175$); ten opzichte van de ruwe EcoLogits-ondergrens kan de verhouding oplopen tot meer dan een factor 100. De ecologische voetafdruk van de lokale opstelling in Italië bedraagt gemiddeld $\approx 21,6$ mgCO₂eq per 1000 tokens, tegenover $\approx 5,8$ mgCO₂eq per 1000 tokens voor de cloud. Beide metingen dragen inherente onzekerheidsmarges: de lokale CodeCarbon-metingen kennen een meetafwijking van tot circa 40%, terwijl de cloud-schattingen van EcoLogits zijn gecorrigeerd via een empirisch bepaalde prefill-factor die op zijn beurt onzekerheden draagt.

De integratie van lokale LLM's in een webapplicatie verhoogt de ecologische voetafdruk per inferentie-aanroep significant, maar de absolute uitstoot per aanroep blijft voor beide opties zeer laag. De architectuurkeuze moet bijgevolg niet uitsluitend op ecologische gronden worden gemaakt; kosten, privacy, latentie en beschikbaarheid spelen een minstens even grote rol.

10.1.1. Deelvragen in het probleemdomain

Wat is het energieverbruik en de CO₂-uitstoot van cloudgebaseerde AI-aanroepen in een webapplicatie? En die van lokaal gehoste AI-modellen?

Voor de documentextractie-toepassingsscenario van dit onderzoek (Italiaanse energiefacturen met gemiddeld ≈ 10.000 inputtokens per aanroep en ≈ 65 outputtokens) bedraagt het gemiddelde energieverbruik per aanroep voor de cloud $\approx 0,16$ Wh (gecorrigeerde bovengrens) en voor lokale modellen $\approx 0,86$ Wh (PUE-gecorrigeerd). Omgerekend naar CO₂eq (Italiaanse emissiefactor 0,285 kgCO₂eq/kWh) produceert een lokale aanroep gemiddeld $\approx 0,22$ gCO₂eq, tegenover $\approx 0,06$ gCO₂eq via de cloud. De cloud scoort energiezuiniger dankzij de hogere hardware-efficiëntie van enterprise-GPU's (NVIDIA H100/A100), serverside batching en een gunstiger PUE van het datacenter. De lokale opstelling op consumentenhardware heeft een hogere energie-intensiteit per token, deels gecompenseerd door 4-bit kwantisatie van de modellen.

Hangt de keuze af van het toepassingsscenario binnen de applicatie? Of is de ene optie beter in alle scenario's?

Het toepassingsscenario is een cruciale variabele. De bevinding dat cloud gemiddeld $\approx 5\times$ efficiënter is per token, geldt specifiek voor documentextractie met een hoge input/output-verhouding ($\approx 150:1$). Bij toepassingsscenario's met een kortere inputprompt (zoals korte chatberichten of enkelvoudige vragen) verkleint de prefill-fase als aandeel van het totale energieverbruik drastisch, waardoor het voordeel van de cloud afneemt. Lokale inferentie kan dan financieel en qua latentie aantrekkelijker worden. Bovendien geldt dat de privacyvereisten van een bedrijf de keuze kunnen mandateren: wanneer documenten bedrijfsgevoelige informatie bevatten, kan cloud-inferentie principieel worden uitgesloten, ongeacht de ecologische balans.

Welke methoden zijn geschikt om de impact te meten van deze AI-modellen?

Dit onderzoek heeft twee complementaire meetmethoden ingezet: **CodeCarbon** voor directe hardware-instrumentatie bij lokale inferentie (via **NVML** voor de **GPU** en **RAPL** voor de **CPU**), en **EcoLogits** voor bottom-up levenscyclusanalyse bij cloud-aanroepen. Beide tools zijn geschikt als startpunt, maar kennen methodologische beperkingen. CodeCarbon heeft een meetafwijking van tot $\approx 40\%$ ten opzichte van directe stroommetingen; EcoLogits onderschat de cloud-energie voor documentextractie-werklasten sterk wanneer de prefill-fase niet wordt gecorrigeerd. Een voorgesteld meetkader combineert beide tools met een expliciete PUE-normalisatie en een prefill-correctiefactor (κ), zodat lokale en cloud-metingen op hetzelfde abstractieniveau worden vergeleken.

10.1.2. Deelvragen in het oplossingsdomein**Hoeveel energie en CO₂-uitstoot genereren beide systemen bij het verwerken van eenzelfde AI-werklast?**

Over 6241 inferentieaanroepen verdeeld over zes modellen en 21 documenten werd gemeten dat de lokale opstelling gemiddeld $\approx 0,86$ Wh per aanroep verbruikt (PUE-gecorrigeerd), tegenover $\approx 0,16$ Wh voor cloud (gecorrigeerde bovengrens). Uitgedrukt in tokens is de verhouding $\approx 3,7$ tot $8,5$ afhankelijk van het model: Llama 3.1 8B en Qwen3-VL 8B sluiten het dichtst aan bij de cloud (factor $\approx 3,7$ tot $3,8$), terwijl Aper-tus 8B lokaal de grootste afwijking toont (factor $\approx 8,5$). De **CO₂**-uitstoot volgt exact hetzelfde patroon, aangezien **CO₂eq** lineair evenredig is met het energieverbruik via de regionale emissiefactor.

Hoe verhouden performantie en energie-efficiëntie zich tussen een lokaal model en een cloud-LLM binnen een webomgeving?

Cloud-inferentie is gemiddeld $\approx 2,3\times$ sneller per aanroep dan lokale inferentie bij dit belastingsprofiel. Dit voordeel concentreert zich volledig in de prefill-fase:

cloud verwerkt inputtokens $\approx 2,2\times$ sneller (gemiddeld 5.835 vs. 2.678 tokens/s). De decode-snelheid (die het meest zichtbaar is voor eindgebruikers in een interactieve interface) is bijna gelijk (≈ 51 vs. 39 tokens/s). Voor batchverwerking, zoals de factuurverwerking bij Turtle Srl, is de prefill-snelheid bepalend; de cloud heeft hier een concreet voordeel. De structurele extractiekwaliteit is voor de meeste modellen hoog ($\geq 97\%$), maar de inhoudsnauwkeurigheid varieert sterk: Llama 3.1 8B scoort het best (62 tot 66/100 op de ground-truth vergelijking), terwijl Apertus 8B lokaal dramatisch faalt (14,9/100). Dit laatste resultaat is opmerkelijk: Apertus is juist expliciet ontworpen voor meertaligheid (15T tokens uit meer dan 1.800 talen, $\approx 40\%$ niet-Engelstalig), wat een eenvoudige verklaring via taalondervertegenwoordiging uitsluit. Een plausibeler mechanisme is dat de Q4_K_M-kwantisatie de niet-standaard architectuurcomponenten van Apertus (xlELU, QK-Norm) suboptimaal ondersteunt, gecombineerd met de recente uitgifte van het model (september 2025) (Apertus e.a., [2025](#)).

Kunnen de resultaten vertaald worden naar praktische richtlijnen voor duurzame AI-integratie in webontwikkeling?

Ja. De bevindingen zijn vertaald naar een beslissingskader (Hoofdstuk 9), een negentraps beslissingsboom dat IT-managers en webontwikkelaars begeleidt op basis van contextuele parameters: dataprivacy, hardwarebeschikbaarheid, aanroepvolume, inputtokenlengte, cloud-tarief, latentievereiste, economische BEP, lokale stroommix en kwaliteitsvalidatie. De kernboodschap luidt: **cloud verdient de ecologische voorkeur** zolang er geen privacybezwaren zijn, het aanroepvolume onder de break-even drempel ligt ($\approx 2,6$ miljoen aanroepen voor de duurdere cloud-modellen) en de datacenterlocatie een relatief groene energiemix hanteert. Lokale inferentie is zinvol wanneer privacy een harde vereiste is, wanneer hoge aanroepvolumes de hardware-investering amortiseren, of wanneer de bedrijfsomgeving over betrouwbare, krachtige hardware beschikt.

10.2. Kritische reflectie en beperkingen

De resultaten van dit onderzoek zijn robuust voor de richting van het verschil (cloud is energie-efficiënter dan lokaal voor deze toepassingsscenario), maar de absolute waarden dragen meerdere onzekerheidslagen. De lokale energiemetingen via CodeCarbon zijn software-gebaseerde schattingen met een gekende foutmarge van tot $\approx 40\%$. De cloud-energieschattingen via EcoLogits zijn afhankelijk van een bottom-up levenscyclusmodel dat de prefill-fase standaard negeert; de toegepaste correctiefactor ($\alpha = 0,175$) is empirisch bepaald op basis van gemiddelden over 149 diverse modellen en draagt bijgevolg model-specifieke onzekerheid. De vergelijking is bovendien geen hardware-neutrale exercitie: de lokale modellen draaien op Q4_K_M-kwantisatie, terwijl de cloud-versies op volledige FP16-precisie werken. Dit is een bewuste keuze die de praktische realiteit weerspiegelt, maar

vergelijkingen van modelkwaliteit moeten met die asymmetrie in acht worden genomen.

Verder zijn netwerkenergie en embodied carbon voor beide paden buiten beschouwing gelaten, en zijn de break-even berekeningen gebaseerd op interne volume-projecties en een gecorrigeerde hardware-kostschatting van Turtle Srl. De conclusies over terugverdientijden zijn bijgevolg bedrijfsspecifiek en niet zonder aanpassing generaliseerbaar.

10.3. Toekomstig onderzoek

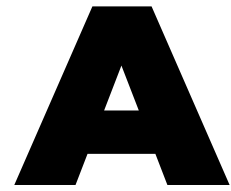
Dit onderzoek opent meerdere paden voor vervolgonderzoek. Ten eerste verdient de **impact van kwantisatieniveau op meertalige extractie, in het bijzonder voor architectureel niet-standaard modellen**, een systematische studie. De dramatische kwaliteitsdaling van Apertus 8B lokaal (Q4_K_M, score 14,9/100) ten opzichte van de cloud-variant (FP16, score 55,1/100) roept vragen op over de oorzaak. Opvallend is dat de technical report van Apertus het model expliciet positioneert als multilinguale prioriteit — voorgetraind op 15T tokens uit meer dan 1.800 talen met $\approx 40\%$ niet-Engelstalig materiaal — waardoor een eenvoudige verklaring via ondervertegenwoordiging van het Italiaans in het trainingskorpus niet volstaat. Een plausibeler mechanisme is dat de niet-standaard architectuurcomponenten van Apertus (xIELU-activatiefunctie, QK-Norm) minder goed worden ondersteund door de GGUF-kwantisatiepijplijn van `llama.cpp`, die primair voor LLaMA-compatibele architecturen is ontwikkeld (Apertus e.a., 2025). Een vergelijking van Q4, Q6, Q8 en FP16 op meertalige extractietaken, gecombineerd met een vergelijking over kwantisatietools, zou hierover uitsluitsel geven.

Ten tweede is er nood aan **directe hardwarevalidatie van energiemetingen**. De 40%-foutmarge van CodeCarbon vormt een structurele zwakte van softwaregebaseerde energieanalyse. Toekomstig onderzoek kan gebruikmaken van inline stroommeters op componentniveau (zoals INA3221-gebaseerde PCIe-power-meters) om de werkelijke GPU-load te meten en CodeCarbon-metingen te kalibreren.

Ten derde is de **prefill-correctiefactor κ** gebaseerd op een beperkte empirische dataset. Een breder onderzoek dat de prefill/decode-energieverdeling rechtstreeks meet op individuele modellen (bij voorkeur via een vLLM-fork met per-request fasemeting) zou nauwkeurigere correctieparameters opleveren en de onzekerheidsbandbreedtes in cloud-energieschattingen verkleinen.

Ten slotte reikt dit onderzoek de methodologie aan, maar de casus is beperkt tot één specifieke toepassingsscenario en infrastructuurcontext. Vervolgonderzoek naar **andere document-toepassingsscenario's** (Bill of Materials, contractanalyse, medische dossiers) op andere hardwareplatforms (ARM-gebaseerde servers, NVIDIA Jetson voor edge-deployments) zou de generaliseerbaarheid van het beslissingskader aanzienlijk vergroten. Evenzo verdient de integratie van een

automatische anonimiseringstap voor privacygevoelige documenten vóór cloud-verwerking aandacht: een dergelijke voorverwerkingsstap zou de cloud-optie toegankelijk maken voor bedrijven die momenteel noodgedwongen voor lokale inferentie kiezen.



Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1. Samenvatting

Steeds meer webapplicaties integreren generatieve AI via cloudgebaseerde Large Language Models (LLM's) zoals ChatGPT en gelijkaardige diensten. Deze modellen bieden krachtige functionaliteit, maar het grootschalige gebruik ervan in datacenters brengt een aanzienlijke ecologische en economische kost met zich mee doordat grote hoeveelheden elektriciteit worden verbruikt en CO₂ wordt uitgestoten. Voor middelgrote softwarebedrijven die AI-functionaliteit in hun toepassingen willen integreren, blijft het onduidelijk of lokaal gehoste LLM's een duurzamer en kostenefficiënter alternatief vormen dan gecentraliseerde generatieve AI-diensten, en onder welke voorwaarden dat het geval is. Deze bachelorproef vertrekt daarom vanuit de onderzoeksvraag: *Hoe beïnvloedt de integratie van Large Language Models (LLM's) in webontwikkelingsframeworks de ecologische voetafdruk in vergelijking met het gebruik van gecentraliseerde generatieve AI-diensten?* Als casus wordt gewerkt met interne prototypes van Turtle Srl voor documentinformatie-extractie en de analyse van industriële Bills of Materials, ondersteund door een geanonimiseerde benchmarkdataset die realistische maar niet-gevoelige testscenario's mogelijk maakt. Het doel van het onderzoek is om een beslissingskader te ontwikkelen dat webontwikkelaars en technische beslissers helpt kiezen tussen lokale, cloudgebaseerde of hybride AI-integratie op basis van ecologische impact, kosten en prestatienoden. Methodologisch wordt een vergelijkende studie uitgevoerd in een proof-of-concept webomgeving waarin dezelfde werkstromen zowel via een lokaal gehost LLM als via een cloudgebaseerde LLM-API worden aangestuurd; voor

de lokale opstelling, die in Italië draait, wordt energieverbruik gemeten met softwarematige tools en vertaald naar CO₂-uitstoot op basis van de Italiaanse elektriciteitsmix, terwijl voor de cloudvariant gebruik wordt gemaakt van token- en gebruiksdata in combinatie met publiek beschikbare emissiefactoren. Op basis van deze metingen worden scenario's uitgewerkt voor verschillende gebruiksvolumes, waarna energieverbruik, geschatte CO₂-uitstoot, responstijd en gebruiksgebonden kosten tussen lokaal en cloud worden vergeleken. Verwacht wordt dat lokaal gehoste LLM's bij hoog en stabiel gebruik per verzoek energie-efficiënter zijn, terwijl cloudoplossingen voordelig blijven bij lage of sterk variabele workloads; het beslissingskader vertaalt deze inzichten naar concrete richtlijnen en een flowchart die de doelgroep ondersteunt bij het maken van onderbouwde, duurzame keuzes in webontwikkeling.

Opmerking

Dit voorstel is gebaseerd op het onderzoeksvoorstel dat werd geschreven in het kader van het vak Research Methods dat ik vorig academiejaar heb uitgewerkt (met medestudent ARNAUD TISON als mede-auteur).

A.2. Inleiding

Webapplicaties maken steeds vaker gebruik van generatieve AI, aangedreven door taalmodellen in de cloud zoals ChatGPT. Deze Large Language Models (LLM's) bieden krachtige functionaliteit voor taken als tekstgeneratie, samenvattingen, semantisch zoeken en code-assistentie. Het grootschalige gebruik in datacenters brengt echter een aanzienlijke ecologische kost met zich mee door het hoge elektriciteitsverbruik en de bijbehorende CO₂-uitstoot. Onderzoek toont aan dat wanneer LLM's intensief in productie worden genomen, de emissies tijdens de inferentiefase de uitstoot van de initiële training kunnen overstijgen. Hierdoor wordt de dagelijkse operationele belasting een cruciale factor in de totale milieu-impact. (Singh e.a., [2024](#))

Voor middelgrote softwarebedrijven die AI willen integreren, is het vaak onduidelijk wat de meest aangewezen strategie is: gebruik blijven maken van cloudgebaseerde diensten of overschakelen naar lokaal gehoste modellen. Het ontbreekt aan inzicht in hoe deze opties zich tot elkaar verhouden op het vlak van ecologische voetafdruk, kosten en prestaties. Cloud-LLM's bieden directe schaalbaarheid en toegang tot zeer grote modellen, maar hun emissieprofiel is sterk gekoppeld aan de infrastructuur en elektriciteitsmix van de gebruikte datacenters. Lokale modellen vragen daarentegen investeringen in hardware en onderhoud, maar bieden de mogelijkheid om duurzamere stroombronnen te benutten en de rekenlast dichterbij de applicatie te plaatsen. De optimale keuze hangt af van contextfactoren zoals het gebruiksvolume, de aard van de taken, latency-eisen en de elektriciteitsmix van de locatie. (Han e.a., [2025](#); Yuan e.a., [2025](#))

Een bijkomend obstakel is dat de ecologische impact in de praktijk moeilijk zichtbaar is. Hoewel er methoden bestaan om energieverbruik en CO₂-uitstoot te schatten, is niet elke aanpak geschikt voor een concrete bedrijfsapplicatie. Bovendien rapporteren cloudproviders zelden directe metingen van het energieverbruik per verzoek, waardoor ontwikkelaars aangewezen zijn op afgeleide indicatoren, zoals het aantal verwerkte tokens. (Krupp e.a., 2025)

De concrete probleemstelling van dit onderzoek wordt benaderd vanuit een specifieke casus bij Turtle Srl, een softwarebedrijf dat bedrijven in hun ecologische en sociale transitie begeleidt door middel van impactmetingen, procesoptimalisatie en technische innovatie. De organisatie staat voor het dilemma hoe AI-functionaliteit voor documentinformatie-extractie (uit pdf's en rapporten) en/of de analyse van industriële Bills of Materials (BOM's) het best kan worden geïmplementeerd. Hoewel deze werkstromen zich lenen voor zowel cloudgebaseerde diensten als lokaal gehoste modellen, ontbreekt het Turtle Srl aan een duidelijk kader om te bepalen welke infrastructuurkeuze voor hun specifieke situatie de laagste ecologische voetafdruk combineert met de vereiste kostenefficiëntie en prestaties. Om dit te onderzoeken stelt het bedrijf een geanonimiseerde benchmarkdataset ter beschikking, gebaseerd op reële documenten en stuklijsten, zodat reproduceerbare tests kunnen worden uitgevoerd zonder vertrouwelijke informatie prijs te geven. De centrale onderzoeksvraag luidt:

Hoe beïnvloedt de integratie van Large Language Models (LLM's) in webontwikkelingsframeworks de ecologische voetafdruk in vergelijking met het gebruik van gecentraliseerde generatieve AI-diensten?

Om deze centrale onderzoeksvraag te beantwoorden, worden deelvragen geformuleerd in zowel het probleemdomein als het oplossingsdomein.

In het *probleemdomein* horen daar de volgende deelvragen bij:

- Wat is het energieverbruik en de CO₂-uitstoot van cloudgebaseerde AI-aanroepen in een webapplicatie? En die van lokaal gehoste AI-modellen?
- Hangt de keuze af van de use case binnen de applicatie? Of is de ene optie beter in alle scenario's?
- Welke methoden zijn geschikt om de impact te meten van deze AI-modellen?

Daarbij wordt gekeken naar energieverbruik per verzoek en per token, geschatte CO₂-uitstoot op basis van landspecifieke elektriciteitsmixen en kwaliteits- en performantieaspecten zoals responstijd en foutgedrag. (Morrison e.a., 2025)

In het *oplossingsdomein* worden volgende deelvragen onderzocht:

- Hoeveel energie en CO₂-uitstoot genereren beide systemen (lokaal en cloud) bij het verwerken van eenzelfde AI-workload?

- Hoe verhouden performantie en energie-efficiëntie zich tussen een lokaal model en een cloud-LLM binnen een webomgeving?
- Kunnen de resultaten vertaald worden naar praktische richtlijnen voor duurzame AI-integratie in webontwikkeling?

Hierbij wordt nadrukkelijk rekening gehouden met implementatiekeuzes, het verwachte gebruiksvolume en de elektriciteitsmix van de infrastructuur, met specifieke aandacht voor Italië (locatie van de proof-of-concept) en België als vergelijkingspunt. (Yuan e.a., 2025)

De doelgroep van dit onderzoek bestaat uit webontwikkelaars, technische beslis-sers en IT-managers in middelgrote softwarebedrijven zoals Turtle Srl, die AI willen integreren met oog voor duurzaamheid, kosten en prestaties. Het beoogde eindresultaat omvat, naast de scriptie, een proof-of-concept webomgeving en een beslissingskader. Dit kader, in de vorm van een flowchart en richtlijnen, moet de doelgroep in staat stellen om, vertrekkend van de eigen use case, profiel, kwaliteitseisen en locatie, een onderbouwde keuze te maken tussen lokale of cloudgebaseerde integratie.

A.3. Literatuurstudie

A.3.1. Duurzaamheid en AI-impact

De impact van AI op duurzaamheid is uitgegroeid tot een belangrijk onderzoeksdomein. Singh e.a. (2024) schetsen een uitgebreid overzicht van de energie- en materiaalimpact van grote taalmodellen en benadrukken dat zowel training, inferentie als infrastructuur bijdragen aan de totale CO₂-voetafdruk. Zij wijzen erop dat bij grootschalig gebruik de emissies van de operationele fase (inferentie) die van de initiële training kunnen overstijgen. (Singh e.a., 2024)

Morrison e.a. (2025) pleiten voor een holistische evaluatie van de milieu-impact, waarbij operationele emissies (operational carbon) worden gecombineerd met emissies verbonden aan productie, transport en afschrijving van hardware (embodied carbon). Volgens hun analyse kan het verkleinen van modellen, het optimaliseren van hardware en het inzetten van energie-efficiënte datacenters de totale impact substantieel reduceren. (Morrison e.a., 2025)

A.3.2. Lokale versus cloudgebaseerde inferentie

Meerdere studies vergelijken lokale en cloudgebaseerde inferentie van LLM's. Han e.a. (2025) concluderen op basis van gesimuleerde keuzestudies dat lokale inferentie, bij herhaald gebruik en voldoende schaal, minder energie per verzoek kan verbruiken dan cloudvarianten. Voorwaarde is wel dat de hardware efficiënt wordt benut en het model goed is afgestemd op de taak. Tegelijkertijd blijven cloudoplossingen aantrekkelijk voor complexe modellen en wisselende workloads. (Han e.a., 2025)

Yuan e.a. (2025) introduceren een kader voor local-cloud inference offloading. Hierbij wordt inferentie dynamisch verdeeld tussen lokale infrastructuur en cloud, afhankelijk van factoren zoals taakcomplexiteit en latency-eisen. Zij rapporteren dat een dergelijke hybride benadering de responstijd kan verlagen en het energieverbruik efficiënter kan verdelen, zonder significante afname van de antwoordkwaliteit. (Yuan e.a., 2025)

A.3.3. Energieverbruik per token en emissie-intensiteit

Verschillende auteurs koppelen het energieverbruik van LLM's rechtstreeks aan het aantal verwerkte tokens. Krupp e.a. (2025) tonen aan dat de energieconsumptie in datacenters voor AI-agents in belangrijke mate wordt bepaald door het tokenvolume en de infrastructuurconfiguratie. Zij pleiten voor transparantere rapportering en standaardmeetmethoden, zodat de milieukost van toepassingen inzichtelijker wordt. (Krupp e.a., 2025)

Voor dit onderzoek is de geografische context essentieel. De proof-of-concept wordt uitgevoerd op infrastructuur in Italië, waardoor de CO₂-uitstoot in eerste instantie wordt berekend aan de hand van de Italiaanse elektriciteitsmix. Recente gegevens tonen dat de gemiddelde emissiefactor van het Italiaanse stroomnet hoger ligt dan in landen met een groter aandeel kernenergie en hernieuwbare bronnen, zoals België. Door de gemeten energiewaarden met zowel de Italiaanse als de Belgische emissiefactor te combineren, kan kwantitatief worden aangetoond hoe sterk de locatie van de infrastructuur de uiteindelijke CO₂-voetafdruk beïnvloedt. (Nowtricity, 2025a, 2025b)

A.3.4. Implementatie- en infrastructuurimpact

Naast het directe energieverbruik spelen ook implementatie- en infrastructuurkeuzes een rol. Het opzetten van lokale infrastructuur vereist hardware, opslag, koeling en netwerkvoorzieningen, elk met een eigen productie- en afschrijvingsprofiel. In cloudscenario's wordt deze infrastructuur gedeeld, maar de concrete emissies blijven vaak minder transparant voor de gebruiker. (Morrison e.a., 2025)

Singh e.a. (2024) benadrukken dat, zeker voor kleinere organisaties, de totale milieu-impact over de volledige levenscyclus moet worden bekeken. Indien een lokaal systeem grotendeels onderbenut blijft, kan de ecologische winst ten opzichte van gedeelde cloudinfrastructuur kleiner zijn dan verwacht. (Singh e.a., 2024)

A.3.5. Meetinstrumenten voor energie en CO₂

Voor het meten en schatten van energieverbruik en CO₂-uitstoot bestaan verschillende tools. CodeCarbon is een veelgebruikte Python-bibliotheek die runtimegegevens over hardwareverbruik koppelt aan landspecifieke emissiefactoren om de uitstoot te schatten. Dergelijke tools zijn geschikt voor lokale omgevingen en inte-

gratie in de backend. (Morrison e.a., 2025)

In een cloudcontext is directe meting vaak niet mogelijk. Onderzoekers maken daarom gebruik van indirecte schattingen door informatie over verwerkte tokens te combineren met gemiddeld energieverbruik per token en externe data over de elektriciteitsmix (Krupp e.a., 2025). In deze bachelorproef worden de resultaten uit de cloud op die manier omgerekend naar energieverbruik en CO₂-uitstoot per verzoek, zodat ze in dezelfde eenheden kunnen worden vergeleken met de lokaal gemeten waarden uit de proof-of-concept.

A.3.6. Samenvatting

Samenvattend laat de literatuur zien dat de ecologische impact van LLM's substantieel is en sterk afhangt van inferentiegebruik en infrastructuurkeuzes. Lokale en cloudgebaseerde opties kennen elk hun eigen voor- en nadelen qua energie, emissies, performantie en beheer. Ook de elektriciteitsmix van de locatie is een bepalende factor. Er is echter nood aan transparantere metingen en praktische kaders om deze trade-offs in concrete casussen te evalueren. Deze bachelorproef speelt hierop in door voor een specifieke webtoepassing bij Turtle Srl de vergelijking experimenteel uit te voeren, rekening houdend met de Italiaanse context en een Belgisch referentiescenario.

A.4. Methodologie

Het onderzoek hanteert een vergelijkende experimentele aanpak op basis van de casus bij Turtle Srl. De kern bestaat uit het opzetten van een proof-of-concept webomgeving waarin interne prototypes voor documentextractie en stuklijst-analyse zowel met een lokaal gehost als met een cloudgebaseerd LLM-systeem worden aangestuurd. Het stappenplan is opgedeeld in zes fasen, gespreid over twaalf weken. Elke fase heeft een duidelijk doel, een aanpak en deliverables.

Fase 1: Literatuur en casusverkenning (week 1-2)

In fase 1 wordt een gerichte literatuurstudie uitgevoerd rond energieverbruik en CO₂-uitstoot van LLM-inferentie, meetmethoden en de rol van locatie en elektriciteitsmix.

Parallel wordt, in overleg met Turtle Srl, de casus verder afgebakend. Er wordt bepaald welke prototypes en functionaliteit representatief zijn en welke geanonimiseerde data gebruikt kan worden. De deliverables zijn een uitgewerkt literatuuroverzicht en een casusomschrijving.

Fase 2: Requirementsanalyse en experimenteel ontwerp (week 3-4)

In fase 2 worden de functionele en niet-functionele requirements voor de testomgeving vastgesteld. Dit gebeurt via gesprekken met de co-promotor en betrok-

kenen bij Turtle Srl om wensen rondom functionaliteit, kwaliteit, privacy en performance te verduidelijken. Op basis hiervan wordt het experimentele ontwerp uitgewerkt. Dit omvat een beschrijving van de AI-taken, de definitie van workloadprofielen (aantal verzoeken, variatie, pieken) en de selectie van te meten indicatoren (energie, CO₂, tijd, fouten). Ook worden afspraken gemaakt over de configuratie om vergelijkbaarheid tussen lokaal en cloud te garanderen. De deliverable is een formeel experimenteel protocol.

Fase 3: Implementatie van de testomgeving (week 5–6)

In fase 3 wordt de proof-of-concept webomgeving opgezet. De nadruk ligt op het reproduceren van de relevante werkstromen. De omgeving bestaat uit een webfrontend voor het uitvoeren van testscenario's en een backend die zowel het lokale model als de cloud-API's kan aanspreken via een uniforme interface. Logging- en meetcomponenten registreren per verzoek alle relevante gegevens. Het lokale model wordt gehost op infrastructuur in Italië; voor de cloudscenario's worden commerciële diensten gebruikt. De deliverable is een werkende testomgeving waarin dezelfde workflow via beide paden kan worden uitgevoerd.

Fase 4: Simulatie en dataverzameling (week 7–9)

In fase 4 worden de gesimuleerde workloads uitgevoerd. Scenario's worden gebaseerd op de benchmarkdataset en de opgestelde profielen. Batches van verzoeken worden naar zowel de lokale als de cloudvariant gestuurd. Per verzoek worden tijdstempels, responstijd en kenmerken van in- en uitvoer gelogd. Voor de lokale omgeving wordt energieverbruik gemeten via software die hardwaregebruik omzet naar energie- en CO₂-schattingen. Voor de cloudomgeving worden het aantal verwerkte tokens en andere metadata geregistreerd, waarna per verzoek een schatting van het energieverbruik wordt gemaakt op basis van literatuurwaarden voor energie per token; deze energie wordt vervolgens, net zoals bij de lokale metingen, omgerekend naar CO₂-uitstoot met een vaste emissiefactor, zodat beide benaderingen in dezelfde eenheden vergelijkbaar zijn. Foutcodes worden bijgehouden om de robuustheid te beoordelen.

Omdat de proof-of-concept in Italië draait, worden de lokale CO₂-schattingen primair berekend met de Italiaanse emissiefactor. Daarnaast wordt een scenario uitgewerkt waarbij dezelfde energiegegevens worden omgerekend met de Belgische emissiefactor. Zo kan onderzocht worden hoe de locatiekeuze de relatieve impact beïnvloedt. De deliverable is een gestructureerde dataset met alle meetwaarden.

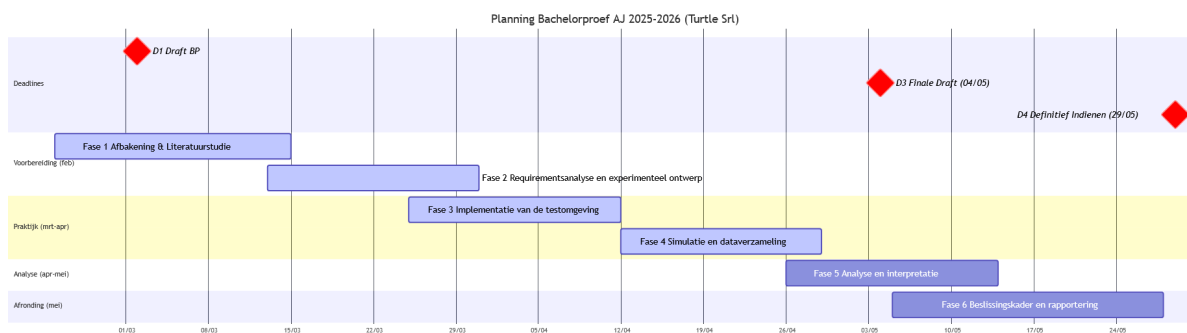
Fase 5: Analyse en interpretatie (week 10–11)

In fase 5 wordt de verzamelde data geanalyseerd. Er worden gemiddelden en spreidingen berekend voor responstijd, energieverbruik en CO₂-uitstoot per taak en scenario (voor zowel de Italiaanse als Belgische factor). Ook worden de kosten

bij verschillende gebruiksvolumes berekend. De analyse focust op het identificeren van patronen: bij welke volumes is lokale inferentie gunstiger en wanneer biedt de cloud voordelen? De invloed van de elektriciteitsmix wordt expliciet meegenomen. De deliverables zijn representatieve tabellen, grafieken en een eerste set conclusies.

Fase 6: Beslissingskader en rapportering (week 12)

In de laatste fase worden de bevindingen vertaald naar een praktisch beslissingskader. Er wordt een flowchart opgesteld die technische beslissers door relevante vragen leidt (gebruiksvolume, latency-eisen, privacy, locatie, duurzaamheidsdoelen). Afhankelijk van de antwoorden volgt een aanbeveling voor lokale of cloudgebaseerde integratie. Tevens wordt de bachelorproef verder uitgewerkt en afgerond. De deliverables zijn het beslissingskader en de afgewerkte thesis. De globale timing van de verschillende fasen van dit onderzoek is samengevat in de Gantt-grafiek in Figuur A.1.



Figuur A.1: Planning bachelorproef 2025–2026

A.5. Verwacht resultaat, conclusie

Op basis van de literatuur en de casusopzet worden enkele duidelijke tendensen verwacht. Voor webtoepassingen met een relatief hoog en voorspelbaar gebruiksvolume lijkt lokale LLM-inferentie per verzoek een lager energieverbruik en lagere CO₂-uitstoot te kunnen realiseren, zeker bij een gematigde emissiefactor. Bij lage of sterk fluctuerende workloads zullen cloudgebaseerde oplossingen naar verwachting gunstiger scoren, door het ontbreken van initiële investeringskosten en standby-verliezen.

Gezien de uitvoering in Italië, is de verwachting dat dezelfde hoeveelheid lokaal verbruikte elektriciteit leidt tot een hogere CO₂-uitstoot dan in een hypothetisch Belgisch scenario. De vergelijkende analyse moet aantonen in welke mate de geografische locatie en stroommix de milieuvoordelen van lokale inferentie versterken of afzwakken. Mogelijk blijkt dat een lokale oplossing in België een duidelijkere CO₂-reductie oplevert dan in Italië, terwijl de kostenverhouding minder sterk wij-

zigt.

Daarnaast wordt verwacht dat lokale integratie in deze casus gemiddeld lagere latency en meer datacontrole biedt, terwijl cloudmodellen wellicht een hogere antwoordkwaliteit leveren bij complexe taken. Door systematische vergelijking ontstaat een genuanceerd beeld van de trade-offs tussen duurzaamheid, kosten, performantie en kwaliteit. Deze inzichten vormen de basis voor het beslissingskader. Het beslissingskader moet de doelgroep in staat stellen om, op basis van specifieke eisen en context, een onderbouwde keuze te maken. Voor Turtle Srl betekent dit dat prototypes niet enkel functioneel, maar ook op ecologische voetafdruk en kostenefficiëntie worden geëvalueerd. Hiermee draagt de bachelorproef bij aan verantwoorde AI-toepassingen en de discussie over duurzame digitalisering.

Bibliografie

- Apertus, P., Hernández-Cano, A., Hägele, A., Huang, A. H., Romanou, A., Solergibert, A.-J., Pasztor, B., Messmer, B., Garbaya, D., Ďurech, E. F., Hakimi, I., Giraldo, J. G., Ismayilzada, M., Foroutan, N., Moalla, S., Chen, T., Sabolčec, V., Xu, Y., Aerni, M., ... Schlag, I. (2025). Apertus: Democratizing Open and Compliant LLMs for Global Language Environments. <https://doi.org/10.48550/ARXIV.2509.14233>
- Artifex Software, Inc. (2024). *pymupdf4llm: PDF to Markdown for LLM ingestion*. GitHub.
- Bai, G., Chai, Z., Ling, C., Wang, S., Lu, J., Zhang, N., Shi, T., Yu, Z., Zhu, M., Zhang, Y., Song, X., Yang, C., Cheng, Y. & Zhao, L. (2024). Beyond Efficiency: A Systematic Survey of Resource-Efficient Large Language Models. <https://doi.org/10.48550/ARXIV.2401.00625>
- Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., ... Zhu, K. (2025). Qwen3-VL Technical Report. <https://doi.org/10.48550/ARXIV.2511.21631>
- Bannour, N., Ghannay, S., Névéol, A. & Ligozat, A.-L. (2021). Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools. In N. S. Moosavi, I. Gurevych, A. Fan, T. Wolf, Y. Hou, A. Marasović & S. Ravi (Red.), *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing* (pp. 11–21). Association for Computational Linguistics. <https://doi.org/10.18653/v1/2021.sustainlp-1.2>
- Berthelot, A., Caron, E., Jay, M. & Lefèvre, L. (2024). Estimating the environmental impact of Generative-AI services using an LCA-based methodology. *Procedia CIRP*, 122, 707–712. <https://doi.org/10.1016/j.procir.2024.01.098>
- Bulian, J., Schäfer, M. S., Amini, A., Lam, H., Ciaramita, M., Gaiarin, B., Hübscher, M. C., Buck, C., Mede, N. G., Leippold, M. & Strauß, N. (2023). Assessing Large Language Models on Climate Information. <https://doi.org/10.48550/ARXIV.2310.02932>
- Caravaca, F., Cuevas, Á. & Cuevas, R. (2025). From Prompts to Power: Measuring the Energy Footprint of LLM Inference. <https://doi.org/10.48550/ARXIV.2511.05597>
- Cho, J. e.a. (2024). Data Center Energy Evaluation Tool Development and Analysis of Power Usage Effectiveness with Different Economizer Types in Various Climate Zones [Open Access via MDPI]. *Buildings*, 14(1), 299. <https://www.mdpi.com/2075-5309/14/1/299>

- Clemm, C., Stobbe, L., Wimalawarne, K. & Druschke, J. (2024). Towards Green AI: Current status and future research. <https://doi.org/10.48550/ARXIV.2407.10237>
- CodeCarbon Contributors. (g.d.). *Methodology*. Verkregen 26 maart 2026, van <https://docs.codecarbon.io/latest/explanation/methodology/>
- Courty, B., Schmidt, V., Goyal-Kamal, MarionCoutarel, Feld, B., Lecourt, J., LiamConnell, SabAmine, inimaz, supatomic, Léval, M., Blanche, L., Cruveiller, A., ouminasara, Zhao, F., Joshi, A., Bogroff, A., Saboni, A., de Lavoreille, H., ... Minerva-Books. (2024). mlco2/codecarbon: v2.4.1. <https://doi.org/10.5281/ZENODO.11171501>
- Courty, B., Schmidt, V., Goyal-Kamal, inimaz, MarionCoutarel, Blanche, L., Feld, B., Lecourt, J., LiamConnell, Saboni, A., LLORET, P., SabAmine, supatomic, Léval, M., cianc, Cruveiller, A., ouminasara, Zhao, F., Bauer, C., ... rosekelly6400. (2026). mlco2/codecarbon: v3.2.6 (Versie v3.2.6). Zenodo. <https://doi.org/10.5281/zenodo.19334697>
- Dong, Y., Ruan, C. F., Cai, Y., Lai, R., Xu, Z., Zhao, Y. & Chen, T. (2024). XGrammar: Flexible and Efficient Structured Generation Engine for Large Language Models. <https://doi.org/10.48550/ARXIV.2411.15100>
- Ember. (2026). Electricity Data Explorer: Carbon Emissions Intensity [Data geraadpleegd via de interactieve Electricity Data Explorer; geselecteerde indicator: CO₂-intensiteit (totale brandstofmix); referentiejaar: 2025]. Verkregen 7 mei 2026, van [https://ember-energy.org/data/electricity-data-explorer/?entity=Belgium&multi=true&data=co2_intensity&fuel=total&entity=Italy&entity=France&entity=Germany&entity=United + States&entity = World&chart = single_year&date=2025-01-01](https://ember-energy.org/data/electricity-data-explorer/?entity=Belgium&multi=true&data=co2_intensity&fuel=total&entity=Italy&entity=France&entity=Germany&entity=United+States&entity=World&chart=single_year&date=2025-01-01)
- European Central Bank. (2026). Euro foreign exchange reference rates. Verkregen 8 mei 2026, van https://www.ecb.europa.eu/stats/policy_and_exchange_rates/euro_reference_exchange_rates/html/index.en.html
- Eurostat. (2022). Electricity prices for non-household consumers - bi-annual data (from 2007 onwards). https://doi.org/10.2908/NRG_PC_205
- Fernandez, J., Na, C., Tiwari, V., Bisk, Y., Luccioni, S. & Strubell, E. (2025). Energy Considerations of Large Language Model Inference and Efficiency Optimizations. <https://doi.org/10.48550/ARXIV.2504.17674>
- Fischer, R. (2025). Ground-Truthing AI Energy Consumption: Validating CodeCarbon Against External Measurements. <https://doi.org/10.48550/ARXIV.2509.22092>
- Fu, Z., Chen, F., Zhou, S., Li, H. & Jiang, L. (2024). LLMCO2: Advancing Accurate Carbon Footprint Prediction for LLM Inferences. <https://doi.org/10.48550/ARXIV.2410.02950>
- GenAI Impact. (g.d.-a). *Environmental Impacts of LLM Inference*. Verkregen 26 maart 2026, van https://ecologits.ai/0.10/methodology/llm_inference/

- GenAI Impact. (g.d.-b). *Tutorial: Providers: Hugging Face Hub*. Verkregen 26 maart 2026, van https://ecologits.ai/latest/tutorial/providers/huggingface_hub/
- Gerganov, G. & contributors. (2023). *llama.cpp: Port of Facebook's LLaMA model in C/C++*. GitHub.
- ggml-org. (2023). *llama.cpp Grammar — GBNF Guide*. Verkregen 16 mei 2026, van <https://github.com/ggml-org/llama.cpp/blob/master/grammars/README.md>
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., ... Ma, Z. (2024). The Llama 3 Herd of Models. <https://doi.org/10.48550/ARXIV.2407.21783>
- Hacker, P. (2023). Sustainable AI Regulation. <https://doi.org/10.48550/ARXIV.2306.00292>
- Han, W., Pawlek, J. & Sivakumar, A. (2025, maart 7). *Evaluating Local and Cloud-Based Large Language Models for Simulating Consumer Choices in Energy Stated Preference Surveys* (onderzoeksrap.). Imperial College London. Verkregen 15 oktober 2025, van <https://arxiv.org/pdf/2503.10652>
- Hugging Face. (2026). Hugging Face Inference Providers. Verkregen 4 mei 2026, van <https://huggingface.co/inference/models>
- Järvenpää, H., Lago, P., Bogner, J., Lewis, G., Muccini, H. & Ozkaya, I. (2023). A Synthesis of Green Architectural Tactics for ML-Enabled Systems. <https://doi.org/10.48550/ARXIV.2312.09610>
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. I., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T. & Sayed, W. E. (2023). Mistral 7B. <https://doi.org/10.48550/ARXIV.2310.06825>
- Jiang, Y., Roy, R. B., Kanakagiri, R. & Tiwari, D. (2025). WaterWise: Co-optimizing Carbon- and Water-Footprint Toward Environmentally Sustainable Cloud Computing. <https://doi.org/10.48550/ARXIV.2501.17944>
- Khan, T., Motie, S., Kocak, S. A. & Raza, S. (2025). Optimizing Large Language Models: Metrics, Energy Efficiency, and Case Study Insights. <https://doi.org/10.48550/ARXIV.2504.06307>
- Kim, Y. G., Gupta, U., McCrabb, A., Son, Y., Bertacco, V., Brooks, D. & Wu, C.-J. (2023). GreenScale: Carbon-Aware Systems for Edge Computing. <https://doi.org/10.48550/ARXIV.2304.00404>
- Krupp, L., Geißler, D., Lukowicz, P. & Karolus, J. (2025). Towards Sustainable Web Agents: A Plea for Transparency and Dedicated Metrics for Energy Consumption. <https://doi.org/10.48550/ARXIV.2502.17903>
- Lacoste, A., Luccioni, A., Schmidt, V. & Dandres, T. (2019). Quantifying the Carbon Emissions of Machine Learning. <https://doi.org/10.48550/ARXIV.1910.09700>

- Lannelongue, L., Grealey, J. & Inouye, M. (2021). Green Algorithms: Quantifying the Carbon Footprint of Computation. *Advanced Science*, 8(12). <https://doi.org/10.1002/advs.202100707>
- Lee, B. C., Brooks, D., van Benthem, A., Gupta, U., Hills, G., Liu, V., Pierce, B., Stewart, C., Strubell, E., Wei, G.-Y., Wierman, A., Yao, Y. & Yu, M. (2024). Carbon Connect: An Ecosystem for Sustainable Computing. <https://doi.org/10.48550/ARXIV.2405.13858>
- Lei, J. e.a. (2022). Prediction of Overall Energy Consumption of Data Centers in Different Locations [Open Access via PubMed Central (PMC)]. *Energies*, 15(11), 3964. <https://pmc.ncbi.nlm.nih.gov/articles/PMC9145178/>
- Li, P., Yang, J., Islam, M. A. & Ren, S. (2023). Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models. <https://doi.org/10.48550/ARXIV.2304.03271>
- Local AI Master Research Team. (2026). *AI VRAM Requirements: How Much GPU Memory for Every Model*. Verkregen 4 mei 2026, van <https://localaimaster.com/blog/vram-requirements-2026>
- Luccioni, A. S., Gamazaychikov, B., Strubell, E., Hooker, S., Jernite, Y. & Wu, C.-J. (2025). AI Energy Score Leaderboard. Verkregen 26 mei 2026, van <https://huggingface.co/spaces/AIEnergyScore/Leaderboard>
- Milutinović, M. & Lukić, M. (2025). HOW GREEN IS YOUR ALGORITHM? ASSESSING THE CARBON FOOTPRINT OF MACHINE LEARNING. *Safety engineering & management - science, industry, education (SEMSIE 2025)*, 87–94. <https://doi.org/10.46793/semsie25.087m>
- Morrison, J., Na, C., Fernandez, J., Dettmers, T., Strubell, E. & Dodge, J. (2025). Holistically Evaluating the Environmental Impact of Creating Language Models. <https://doi.org/10.48550/ARXIV.2503.05804>
- Nowtricity. (2025a). *CO₂ Emissions per kWh in Belgium*. Verkregen 8 december 2025, van <https://www.nowtricity.com/country/belgium/>
- Nowtricity. (2025b). *CO₂ Emissions per kWh in Italy*. Verkregen 20 oktober 2025, van <https://www.nowtricity.com/country/italy/>
- NVIDIA, : Basant, A., Khairnar, A., Paithankar, A., Khattar, A., Renduchintala, A., Malte, A., Bercovich, A., Hazare, A., Rico, A., Ficek, A., Kondratenko, A., Shaposhnikov, A., Bukharin, A., Taghibakhshi, A., Barton, A., Mahabaleshwarkar, A. S., Shen, A., ... Chen, Z. (2025). NVIDIA Nemotron Nano 2: An Accurate and Efficient Hybrid Mamba-Transformer Reasoning Model. <https://doi.org/10.48550/ARXIV.2508.14444>
- NVIDIA Corporation. (2020). *NVIDIA Ampere GA102 GPU Architecture Whitepaper* [Geverifieerde bron voor de Ampere-architectuur die aan de basis ligt van de GeForce RTX 30-serie.]. Verkregen 21 april 2026, van <https://www.nvidia.com>

- [com/content/dam/en-zz/Solutions/geforce/ampere/pdf/NVIDIA-ampere-GA102-GPU-Architecture-Whitepaper-V1.pdf](https://www.nvidia.com/content/dam/en-zz/Solutions/geforce/ampere/pdf/NVIDIA-ampere-GA102-GPU-Architecture-Whitepaper-V1.pdf)
- Phillips, N. (2022, oktober 26). Beyond PUE: A New Class of Data Center Metrics. Verkregen 8 mei 2026, van <https://www.solunacomputing.com/wp-content/uploads/2022/10/Beyond-PUE-Soluna-White-Paper-1.pdf>
- Pydantic Developers. (2025). *Pydantic Documentation*. Verkregen 1 mei 2026, van <https://docs.pydantic.dev/>
- Qi, S., Milojevic, D., Bash, C. & Pasricha, S. (2023). SHIELD: Sustainable Hybrid Evolutionary Learning Framework for Carbon, Wastewater, and Energy-Aware Data Center Management. <https://doi.org/10.48550/ARXIV.2308.13086>
- Rincé, S. & Banse, A. (2025a, november 13). Announcing major improvements in our methodology. Verkregen 11 mei 2026, van <https://ecologits.ai/latest/blog/2025/11/13/major-methodology-update-2025/>
- Rincé, S. & Banse, A. (2025b). EcoLogits: Evaluating the Environmental Impacts of Generative AI. *Journal of Open Source Software*, 10(111), 7471. <https://doi.org/10.21105/joss.07471>
- Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D. & Gadepally, V. (2023). From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. <https://doi.org/10.48550/ARXIV.2310.03003>
- Sapkota, R., Raza, S. & Karkee, M. (2025). Comprehensive Analysis of Transparency and Accessibility of ChatGPT, DeepSeek, And other SoTA Large Language Models. <https://doi.org/10.48550/ARXIV.2502.18505>
- Shumba, N., Tshekiso, O., Li, P., Fanti, G. & Ren, S. (2024). A Water Efficiency Dataset for African Data Centers. <https://doi.org/10.48550/ARXIV.2412.03716>
- Singh, A., Patel, N. P., Ehtesham, A., Kumar, S. & Khoei, T. T. (2024). A Survey of Sustainability in Large Language Models: Applications, Economics, and Challenges. <https://doi.org/10.48550/ARXIV.2412.04782>
- Song, Y., Mi, Z., Xie, H. & Chen, H. (2023). PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU. <https://doi.org/10.48550/ARXIV.2312.12456>
- TechPowerUp. (2020). *NVIDIA GeForce RTX 3070 Specifications*. Verkregen 12 mei 2026, van <https://www.techpowerup.com/gpu-specs/geforce-rtx-3070.c3674>
- Turtle-Srl. (2026). ViVACE. Verkregen 25 maart 2026, van <https://www.turtlesrl.com/en/vivace-model/>
- Wan, Z., Wang, X., Liu, C., Alam, S., Zheng, Y., Liu, J., Qu, Z., Yan, S., Zhu, Y., Zhang, Q., Chowdhury, M. & Zhang, M. (2023). Efficient Large Language Models: A Survey. <https://doi.org/10.48550/ARXIV.2312.03863>

- Wang, X., Du, H., Gao, Y. & Kim, D. I. (2025). AOLO: Analysis and Optimization For Low-Carbon Oriented Wireless Large Language Model Services. <https://doi.org/10.48550/ARXIV.2503.04418>
- Wu, C.-J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F. A., Huang, J., Bai, C., Gschwind, M., Gupta, A., Ott, M., Melnikov, A., Candido, S., Brooks, D., Chauhan, G., Lee, B., Lee, H.-H. S., ... Hazelwood, K. (2021). Sustainable AI: Environmental Implications, Challenges and Opportunities. <https://doi.org/10.48550/ARXIV.2111.00364>
- Xu, M., Cai, D., Wu, Y., Li, X. & Wang, S. (2023). FwdLLM: Efficient FedLLM using Forward Gradient. <https://doi.org/10.48550/ARXIV.2308.13894>
- Xu, M., Yin, W., Cai, D., Yi, R., Xu, D., Wang, Q., Wu, B., Zhao, Y., Yang, C., Wang, S., Zhang, Q., Lu, Z., Zhang, L., Wang, S., Li, Y., Liu, Y., Jin, X. & Liu, X. (2024). A Survey of Resource-efficient LLM and Multimodal Foundation Models. <https://doi.org/10.48550/ARXIV.2401.08092>
- Yuan, L., Han, D.-J., Wang, S. & Brinton, C. G. (2025, februari 18). *Local-Cloud Inference Offloading for LLMs in Multi-Modal, Multi-Task, Multi-Dialogue Settings* (onderzoeksrap.). Purdue University; Yonsei University; IBM T.J. Watson Research Center. Verkregen 15 oktober 2025, van <https://arxiv.org/pdf/2502.11007>