AI and ML

Cover photo by Conny Schneider https://unsplash.com/@choys_

- Argilla
- LLM Models
  - Gemini 2.5 Series
  - Anthropic
  - Claude 4
  - Anthropic Claude 3 Opus
- SLMs
  - Less is More: Recursive Reasoning with Tiny Networks
  - Mistral Small 3.2
  - Hierarchical Reasoning Model
- Claude Code
- OpenCode
- Evaluating AGENTS.md: Are Repository-Level Context Files Helpful for Coding Agents?
- Gemma 4
- AI Agent Resources
  - AI Agent Resources

ML Introduction

Welcome to my Machine Learning and AI notebook.

? AI and ML Fundamentals

- AI Best Practices - some thoughts and principles that I've built up during my career building data projects.
- Learning with Limited Data - thinking around
- AI Causing Chaos - a compilation of examples of AI cautionary tales.

?? Data Collection and Annotation

- Working with Argilla

? LLMs and Small/Specialised Language Models (SLMs)

- Local LLMs
- Reproducibility
- SLMs

ML Best Practices

Machine learning is a complex and multifaceted activity that requires the combination of a number of success factors in order to work. In order to execute machine learning well, it is important to have a good understanding of the processes and variables that frequently come into play together. Here I document and lay out some of the best practices and ML processes that I have learned and continue to learn during my career.

A personal bugbear of mine is when people refer to machine learning as "an art rather than a science". It is true that machine learning involves random-seeming aspects that could be conflated with magic or wizardry. There are a lot of ways that poorly thought-through or executed machine learning projects can go wrong. However, there are well established protocols and principles that can guide a project towards success and we can execute projects in a risk-averse, incremental way allowing us to fail fast before a large amount of time and energy has been spent on a project.

Key Principles

Garbage In; Garbage Out

Machine learning is all about trying to infer statistical relationships between data. There are a lot of reasons why this might not work well which we will explore in depth. It's always important to remember that if you have bad data or a flawed hypothesis, your model is not going to perform well. Conversely, if your model is performing really well (perhaps "*too good to be true*" *levels of well*) this can be a red flag that something is wrong with your data or your process which leads into...

Professional Pessimism

I love this phrase that I picked up during my career stint as a QA Engineer. If something appears to be too good to be true, it probably is. Always question your results and double check your working. Typically if a machine learning model is achieving high-90% in appropriate performance metrics, it is worth your time to investigate whether something weird is happening. You may have training data in your test set (i.e. your model got hold of the exam answer sheet) or your model may have found a short cut. For example, if all the positive examples of fractured ribs come from x-rays of women and all the negative examples come from x-rays of men then the model may have just learned to separate male and female anatomy rather than what a fractured rib looks like.

Keep it Simple, Stupid (KISS)

As tempting as it might be to want to use the latest and greatest ML models, always start with the simplest approach possible and disqualify these approaches based on sound theoretical rationale or empirical experimentation:

- The more complex a modelling approach, the more uncertainty we invite into the process.
- Complex models are significantly more expensive to train and run and scaling them to run quickly on a large amount of data is a challenge.

Keep Receipts

Machine Learning is usually an iterative process. As you move through this process it is important to understand where you came from and why at all times and to be able to refer back to earlier results and hypotheses. Furthermore, in a business setting, you will likely be asked questions about how work that you did days or weeks ago compares to your current iteration of thinking.

Keeping receipts is about using simple tools (good note taking and ticket hygiene) in combination with more specific purpose-built AI and ML tooling to ensure that you can answer these questions at all times.

Deep Learning Best Practices

For deep learning I like to work with torch. My opinion on how torch code should be written aligns quite closely with this styleguide

Model Quantization

Deploying models that are performant (obviously statistically but in this context I primarily mean ***computationally***) is challenging when you are working with large models such as BERT etc.

Quantization involves compressing model weights into smaller, more efficient representations. Weights are normally stored as 32 bit floating point numbers but they can be compressed into 8 bit integers with a very small amount of performance loss.

This article talks about how to do quantization effectively (mirror).

Quantization with Optimum and OpenVino

Openvino is an open source framework from Intel that provides quantization and x86 CPU support for torch and huggingface transformers.

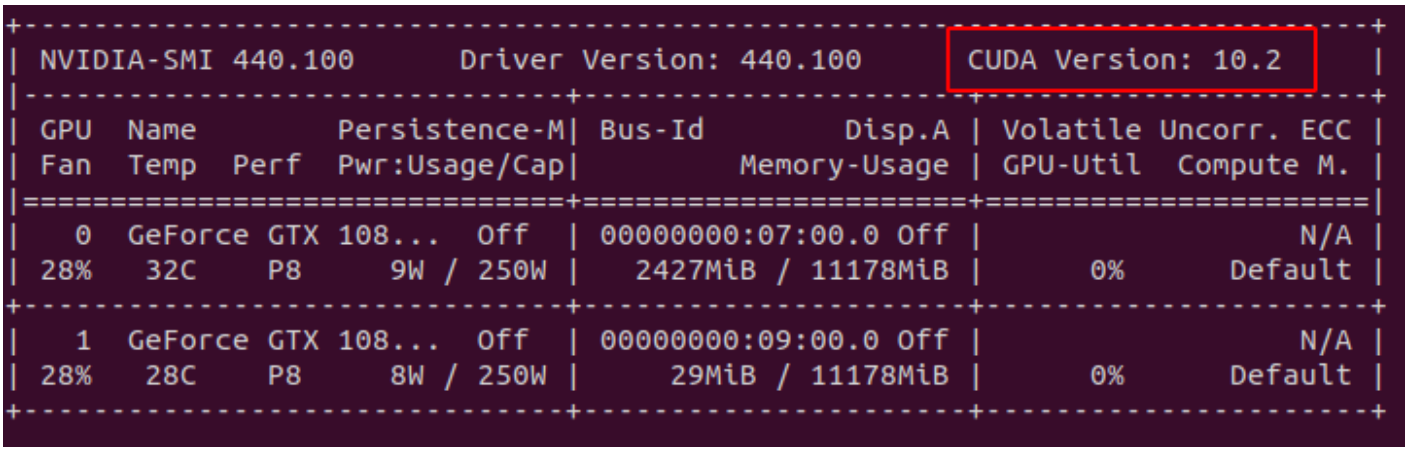
SpaCy GPU

Set Up Environment

It's relatively easy to use SpaCy with a GPU these days.

First set up your conda environment and install cudatoolkit (use nvidia-smi to match versions of the tookit with the drivers):

Run `nvidia-smi`:



Create conda env:

```
conda create -n test python=3.8
conda activate test
conda install pytorch cudatoolkit=10.2 -c pytorch
```

Installing SpaCy

Now install spacy - depending on how you like to manage your python environments either carry on using conda for everything or switch to your preferred package manager at this point.

```
conda install -c conda-forge spacy cupy
```

or

```
pdm add 'spacy[cuda-autodetect]'
```

Download Models

Download a spacy transformer model to make use of your GPU/CUDA setup:

```
python -m spacy download en_core_web_trf
```

Using GPU

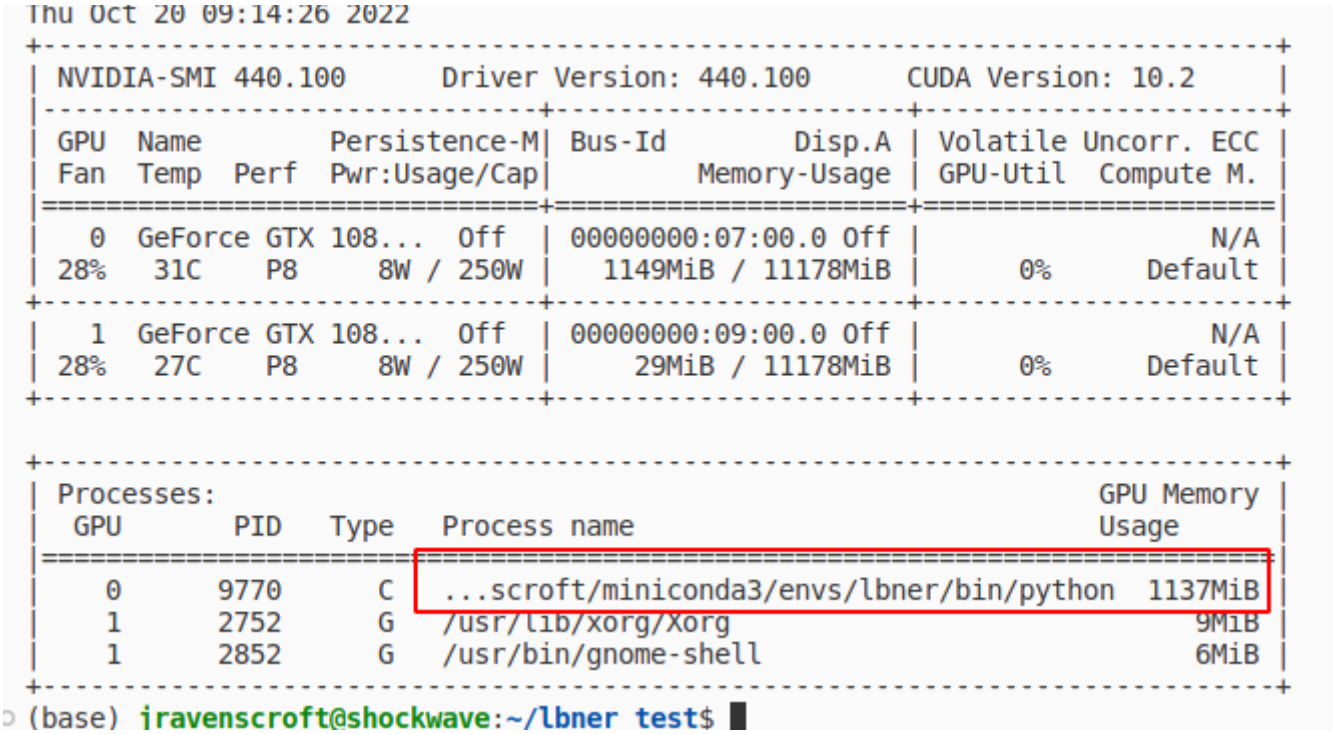
As soon as your code loads you should use the `prefer_gpu()` or `require_gpu()` functions to tell spacy to load cupy then load your model:

```
import spacy


spacy.require_gpu()


nlp = spacy.load('en_core_web_trf')
```

Now you can use the model to do some stuff

```
doc = nlp("My name is Wolfgang and I live in Berlin")


for ent in doc.ents:
    print(ent.text, ent.label_)
```

You can check that the GPU is actually in use with `nvidia-smi`:



Also if you try to use transformer models without a GPU it will hang for AGES and max out your CPUs - another tell that something's not quite right.

SpaCy CoRef

Spacy Coref is an experimental coreference resolution model in spacy

The project repository is here.

There is currently a hard dependency on the LDC OntoNotes dataset which makes it difficult to use without spending money. Hopefully they will release a pre-trained model soon

Core Scientific Concepts (CoreSC)

Core Scientific Concepts (CoreSC) is an annotation scheme used to delineate different parts of scientific discourse in a scientific paper.

There are 11 categories:

- Background
- Conclusion
- Experiment
- Goal
- Hypothesis
- Method
- Model
- Motivation
- Object
- Observation
- Result

Classification Methods

- The original model used for automatic classification of sentences in scientific papers into CoreSCs is SAPIENTA
- Subsequent work by Brack et al incorporates multi-task learning and their models are available here

References

- Guidelines for the Multi-Annotation of Core Scientific Concepts (CoreSC)

Stratified Sampling in Pandas

- Use groupby on the label column to create sub-frames for each label and then use the sample() function.
- Passing an integer gives an exact sample (e.g. `sample(5)` gives 5 rows).
- Passing `frac=0.1` gives a percentage (i.e. 10%)
- Remember to set random_state for reproducible results

```
df = pd.read_csv("path/to/data.csv")


df.groupby('Category', group_keys=False).apply(lambda x: x.sample(frac=0.1, random_state=42))
```

From Crowd Ratings to Predictive Models of Newsworthiness to Support Science Journalism

Paper Link

Authors:

- Sachita Nishal
- Nicholas Diakopoulos

Notes

- Their work comes at the problem from the scientific paper - essentially they are trying to predict whether or not a scientific article might make an interesting news article (as opposed to our work which analyses the news article to attempt to understand which part of the scientific paper is worth mentioning)
- Their definition of newsworthiness is broad and incorporates scientific and social impact but also factors in stuff that we would probably consider less helpful for understanding impact like how controversial the work is.
- They use computer science pre-prints gathered from ArXiv via their API (filtered by category/subject area).
- Their data set is 50 papers in 'train' set, 55 in validation set
- Scientific journalists were paid as experts. Non-specialists were recruited for crowd-sourcing at larger scale (MTurk)
- Crowd-source workers asked to score papers based on the abstract (full text was optionally available but only for gathering additional context/if necessary).
- Workers asked to score along 4 dimensions: actuality, surprise, impact magnitude, impact valence
- They use Likert scales for crowd-rating of newsworthiness
- Crowd source workers achieve moderate associations with the "expert" annotators.
- They train an extra trees random forest classifier with Sentence-BERT feature input (single 768 dim vector used for each article abstract). Additional features include binary indicator if article makes code or data available (hypothesis is that this could increase impact)

- They rank outputs based on likelihood score from the classifier (Extra Trees classifier) and achieve reasonable results

Stable Diffusion

Web UI API

Run the web UI with:

`./webui.sh --api`

Example Python Script

```
import webuiapi



# create API client
api = webuiapi.WebUIApi()

# create API client with custom host, port
#api = webuiapi.WebUIApi(host='127.0.0.1', port=7860)

# create API client with custom host, port and https
#api = webuiapi.WebUIApi(host='webui.example.com', port=443, use_https=True)

# create API client with default sampler, steps.
#api = webuiapi.WebUIApi(sampler='Euler a', steps=20)

# optionally set username, password when --api-auth=username:password is set on webui.
# username, password are not protected and can be derived easily if the communication channel
is not encrypted.
# you can also pass username, password to the WebUIApi constructor.

result1 = api.txt2img(width=512,height=768,
                      prompt="Morgan Freeman in star wars jedi robe",
                      negative_prompt="ugly, out of frame",
                      seed=1003,
                      styles=["anime"],
                      cfg_scale=7,
                      steps=30,
```

```
#                       sampler_index='DDIM',
#                       steps=30,
#                       enable_hr=True,
#                       hr_scale=2,
#                       hr_upscaler=webuiapi.HiResUpscaler.Latent,
#                       hr_second_pass_steps=20,
#                       hr_resize_x=1536,
#                       hr_resize_y=1024,
#                       denoising_strength=0.4,
                      alwayson_scripts = {
                          "ADetailer": {
                          "args": [
                              {
                              "ad_model": "face_yolov8n.pt"
                              }
                          ]
                          }
                      }
                      )

result1.image.save("out.png")
```

AI Causing Chaos

> So much AI turns out to be low-waged people in a call center in the Global South pretending to be robots that Indian techies have a joke about it: “AI stands for ‘absent Indian’” - Cory Doctorow

I am collecting examples of AI fails by people who either deliberately or naively mislead about the capabilities of AI.

2024

- **11/02/24 -** System set up to detect cheating students found 97% of students were cheating and UK officials just accepted this
- **09/02/24 -** Gemini LLM conflates unsafe in the C# memory management context with unsafe as in unethical.
- **05/02/24** - Using neural networks to generate images of fake ids for online verification
- **04/02/24** - via Yoav Goldberg - GEM using models to predict protected characteristics of job applicants when such information is not supplied
- **28/01/24** Dudesy Podcast claimed that they trained an AI to impersonate the late comedian George Carlin but Cory Doctorow and Simon Willison both call bullshit. IT stinks of the "We made an AI watch 100 hours of X..." meme that went round a few years ago.
- **20/1/24** DPD introduce a ChatGPT-powered bot which users jailbreak and get to swear and call DPD 'useless'

2023

- **20/12/23** Chevrolet integrated a ChatGPT-powered bot into their site but because there is no such thing as input sanitization for LLMS people used prompt injection attacks to make the bot agree to sell users cars for $1.
- **24/10/23** Cruise robo-taxi dragged a person 20ft along the floor and had their license to operate in San Francisco suspended.

- **17/01/23** it emerged that a 2016 demo of Tesla's self-driving capabilities was completely staged
-

Tasks

Tasks within ML and NLP

Question Answering

Approaches

Fine-Tuning Sentence-BERT for Question Answering

CapitalOne produced a tutorial (mirror) about using sentence-transformers for Question Answering.

- They use SBERT because it is optimised for fast compute on individual sentence and has good general performance on a number of NLP tasks. They are most interested in the STS capability of SBERT
- They fine-tuned the model on 7k utterances and saw a huge improvement in performance:
  - 52% match rate on 200 test Q-A pairs with no fine tuning
  - 79% match rate on the 200 sample dataset after fine tuning
- They use triplet loss (minimise distance between alike sentences, maximise distance between different sentences)
- This is a nice simple approach where you want to present the answer, as-written, back to the user but its not appropriate for factoid type use cases where we want to highlight the exact word or phrase in the answer text.

Haystack

Haystack is an open source NLP framework for use cases involving large collections of documents. It could be used for searching and ranking type use cases and question answering type use cases.

Haystack works flexibly with existing document stores including DB systems and ElasticSearch.

Coreference Resolution

Co-reference Resolution (CR) is the task of deciding whether two entity mentions refer to the same instance or not.

For example in:

> Joe Biden appeared at the event at 8pm. The president was wearing a Louis Vuitton Tuxedo.

The objective is to identify that Joe Biden and The president are the same entity.

Coreference Detection is related to Relationship Extraction (RE) - in fact you could even say that CR is a special case of RE in the sense that we are interested in the special relationship between entity mentions when they both refer to the same entity.

In-Document Coreference Resolution

This is the "normal" CR case in which you're trying to resolve mentions of entities within the same document e.g. a single news article.

Approaches

- **2022-10-23** A recent blog post from explosion / spaCy shows how they have implemented end-to-end CR in their excellent NLP pipeline but as of writing they do not provide a trained model and they require you to have a copy of the Ontonotes dataset.

Cross-Document Coreference Resolution

Cross-Document Coreference Resolution (CDCR) is when you try to link named entity references across multiple input documents. A use case might be identifying that a number of news articles do actually refer to the same person (e.g. "Joe Biden", "The President").

CDCR is challenging because there are so many possible entities and thus $O(n^2)$ comparisons to make between candidates.

Approaches

- In 2021 we proposed CD^2CR - a CDCR approach across documents and domains that allows us to match mentions of people, places, technologies etc across scientific papers and news articles that discuss them.

Keyword Extraction

Graph-Based Keyword Extraction

Graph-based approaches like TextRank allows the extraction of keywords + phrases based on their centrality to the semantics of the other words in the document.

- https://github.com/SkBlaz/rakun2 - RaKUn 2.0, a very fast keyphrase extraction algorithm suitable for large-scale keyphrase retrieval. It has an associated preprint.

Relationship Extraction

Relationship Extraction (RE) is a task that is related to Coreference Resolution but with a focus on identifying relationships between entities.

In the following example:

> “James, the CTO at Filament AI, lives in the South of England.

We want to identify the following relationships:

```
(James, isCTOof, Filament AI)
(James, livesIn, England)
```

Approaches

- FewRel by the NLP group at Tsinghua University propose a few-shot model which takes 2 entity mentions, in context, and N possible relationship types - it predicts which of the relationships is most likely to apply. It can also predict when none of the relationships are applicable.

Federated Learning

- Flower is a federated learning framework with compatibility with Torch, Tensorflow and others

Large Scale Multi-Label Learning

The Keras website has a tutorial on how to do multi-label learning with a large number of labels:

- https://keras.io/examples/nlp/multi_label_classification/ (mirror)

Machine Learning with Limited Data

Pattern Exploitative Training

PET or Pattern Exploitative Training

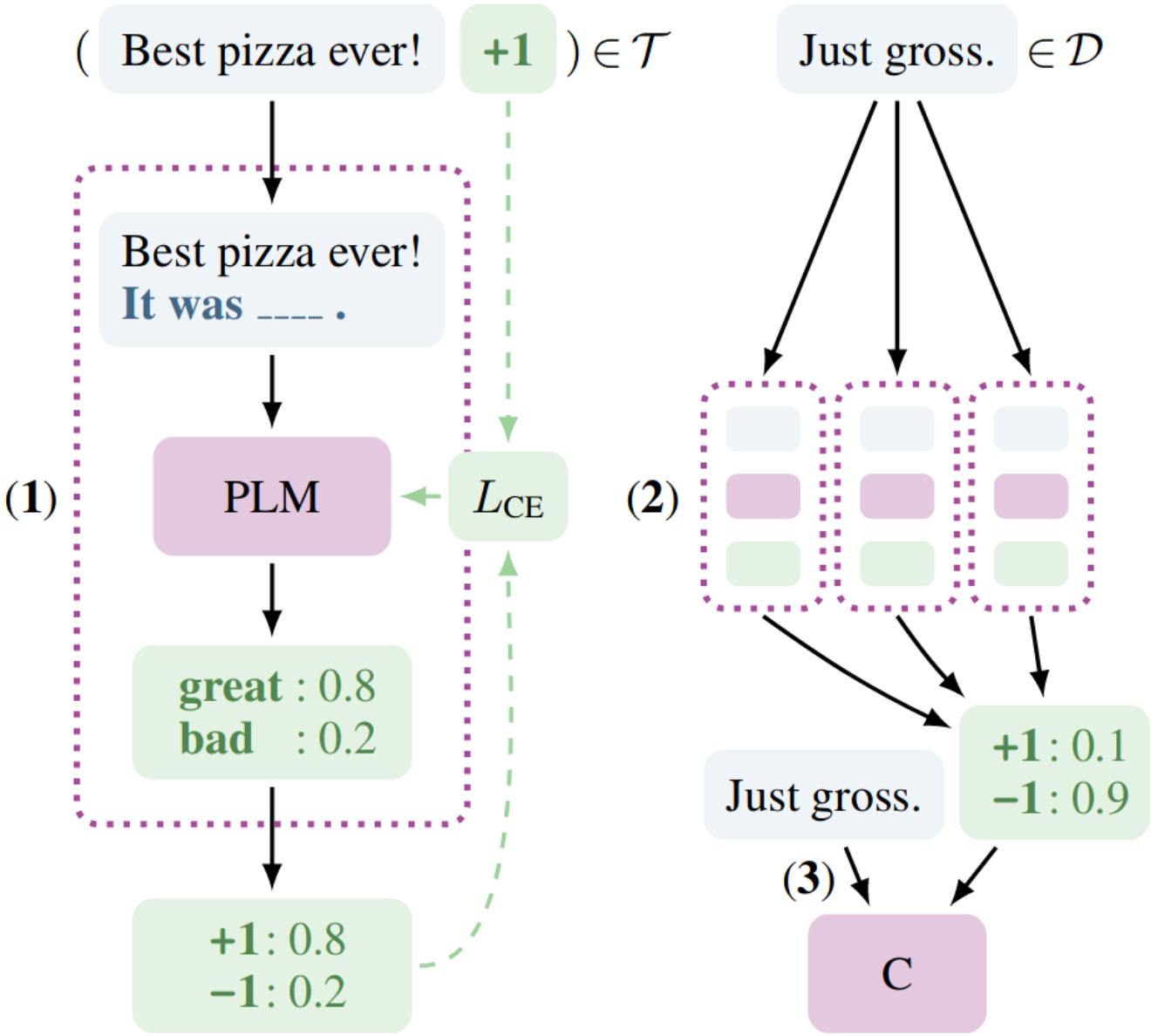


Figure 1: PET for sentiment classification. **(1)** A number of patterns encoding some form of task description are created to convert training examples to cloze questions; for each pattern, a pretrained language model is finetuned. **(2)** The ensemble of trained models annotates unlabeled data. **(3)** A classifier is trained on the resulting soft-labeled dataset.

```
@article{schick2020exploiting,
  title={Exploiting Cloze Questions for Few-Shot Text Classification and Natural Language Inference},
```

```
  author={Timo Schick and Hinrich Schütze},
  journal={Computing Research Repository},
  volume={arXiv:2001.07676},
  url={http://arxiv.org/abs/2001.07676},
  year={2020}
}

@article{schick2020small,
  title={It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners},
  author={Timo Schick and Hinrich Schütze},
  journal={Computing Research Repository},
  volume={arXiv:2009.07118},
  url={http://arxiv.org/abs/2009.07118},
  year={2020}
}
```

Learning with Limited Data

Good machine learning is heavily dependent on good data. A few more good data-points is likely to be worth billions of model parameters.

However, sometimes we need to train models when data is limited. There are a number of strategies that we can try.

Zero-Shot and Few-Shot Learning

- Pattern Exploitative Training is a way to use a small number of examples to train text classifiers. It is technically an example of synthetic data generation.

In Context Learning (ICL)

Synthetic Data Generation and Augmentation

DeBERTa Zero Shot

The DeBERTa zero-shot model is an NLP zero-shot classifier trained by Moritz Laurer and made publically available on HuggingFace

Explainability and Model Analysis

Explainability

Explainability is a big challenge in machine learning.

I wrote a blog post about the ELI5 library and how it can be applied to NLP models.

Introducing ML customers to explainability early on can be a great way to build trust. A colleague suggests using Streamlit tools to allow customers to play with models and understand the contribution different features have had to a particular decision.

Model Confidence Scores

Many ML classification models can provide a confidence score which tells the user how confident the model is that it has made the correct choice.

The values of these confidence scores and what constitutes a "good" or "bad" score can vary a lot depending on the type and behaviour of the model. We often get asked why a particular model only ever seems to be 20% confident when a different model gives 99% confidence. Here's why that happens.

Confidence vs Certainty

Confidence and certainty are related but distinct concepts. We want models to be confident that they are correct (a high score is allocated to one of the labels and a low score allocated to the remaining labels) but also we want it to be certain it is correct (we want it to rely on features/cues that lead to the actual correct outcome).

Neural networks are known to often be over-confident in their results but still incorrect due to the way that modern deep learning learns. The paper On Calibration of Neural Networks dives into this further.

In order for us to be able to trust confidence scores, models must also be well calibrated.

Confidence in Random Forest Models

Random Forest Models are made up of an ensemble of decision tree models which are trained based on randomly selected sub-samples of the full training set - allowing the trees to learn different feature priorities based on the variance in the data that they are "assigned".

A single decision tree model cannot easily tell you how confident it is - the data is passed in and the algorithm traverses the branches in the tree until it reaches a decision. Within the random forest, confidence is calculated by assigning each tree a "vote" on the outcome class and then working out the distributions of votes across the possible outcome classes as a percentage.

> In scikit-learn decision trees[do have confidence scores](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.predict_proba) calculated as the proportion of each class that ended on a given leaf node during training

Say we train a random forest model containing 100 trees on a company sector/industry classification problem with 5 classes. In theory, some of the trees will learn to prioritise the most important features in the dataset. Likewise, we can assume that some of the trees will be trained on less representative sub-samples of the training data and will prioritise less discriminative features.

When we predict on an unseen data sample we might get an output like this:

> We specialise in using AI to improve user experience for customers of high end grocery stores

35 Trees votes for "Consumer Goods"
25 Trees voted for "IT & Technology"
22 Trees voted for "Health & Beauty"
12 Trees voted for "Retail"
6 Trees voted for "Automotive Manufacturing"

I used 100 random trees in my model for easy maths, so we would say in this case that the model is 35% confident that this description is from a company in consumer goods.

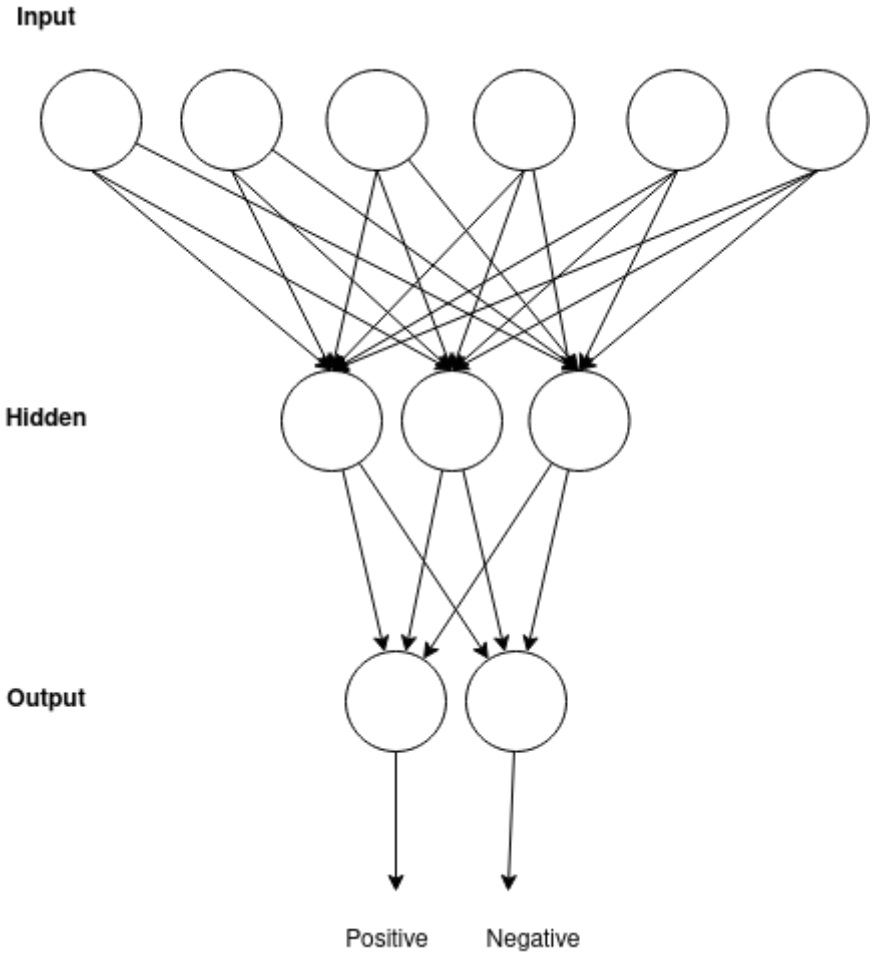
Can We Calibrate Confidence in Random Forest Models?

Confidence calibration is a technique that allows us to more closely map the confidence of an ML model (via its decision function) onto the real life probability that a sample belongs to a particular class.

- Some work has been done on calibration of random forests for probability
- Scikit-Learn provide an out of the box class for providing this functionality CalibratedClassifierCV

Confidence in Neural Networks

Neural classification models can have many different internal structures but tend to have a set of inputs that correspond to the feature vector of the task being learned (e.g. a sparse bag-of-words, a set of RGB pixel values or numerical sensor readings) and a set of outputs equal to the number of classes being predicted.



Activation functions in hidden layers can produce wildly different and un-normalised values depending on the inputs, random initialisations and what is learned (although we can use normalisation constraints in our learning to prevent individual weights from going too far).

The output of the model is therefore typically normalised using SoftMax which essentially takes a weighted average of the results - its a bit like saying "which output received a higher percentage of the overall signal propagated through the network?"

Can We Calibrate Confidence in Neural Networks?

- Some research has been done on this subject

Argilla

Argilla is a data annotation tool with an API that allows you to use external models to automatically suggest labels and carry out active learning.

Resources

- Argilla Few Shot Learning Tutorial - warning some of the code in this notebook is wrong

LLM Models

A list of models

LLM Models

Gemini 2.5 Series

Developer/Trainer	Google
Homepage	https://deepmind.google/models/gemini/pro/

Anthropic

Anthropic is a silicon valley startup and AI company that develops and sells LLM models and LLM-based products and services. They also developed the Model Context Protocol for passing information between models.

Key Information

- Founded as a competitor to OpenAI
- Developed the Claude model series, including Claude 3 Opus, Sonnet, and Haiku
- Known for safety-focused AI development approaches
- Created the Model Context Protocol (MCP) standard

Relationship to Other Models

Anthropic are one of the earliest competitors to OpenAI and their Claude Sonnet and Opus series models were early competitors to GPT-4.

Claude Models

The Claude model series includes several versions:

Claude 3 Series

- Anthropic Claude 3 Opus
- Anthropic Claude 3 Sonnet
- Anthropic Claude 3 Haiku

Other Models

- Anthropic Claude 2
- Anthropic Claude 1

Related Content

- Model Context Protocol

LLM Models

Claude 4

Claude 4 is a large language model developed by Anthropic.

Key Features

- Advanced reasoning capabilities
- Improved safety measures
- Better performance on complex tasks

LLM Models

Anthropic Claude 3 Opus

Overview

Claude 3 Opus is Anthropic's most powerful model, designed for highly complex tasks that require the highest levels of reasoning and understanding.

Capabilities

- Advanced reasoning and problem-solving
- Complex content creation and editing
- Comprehensive understanding of nuanced topics
- High-quality code generation and debugging

Performance

- More capable than Claude 3 Sonnet and Haiku
- Excellent at handling complex reasoning tasks
- Strong performance in benchmarks like MMLU and GSM8K

Use Cases

- Research and analysis
- Complex content creation
- Code generation with extensive reasoning

```
flowchart LR


A[Hard] -->|Text| B(Round)
B --> C{Decision}
C -->|One| D[Result 1]
C -->|Two| E[Result 2]
```

SLMs

Specialised and Small Language Models (SLMs) including information on alternative approaches to transformers

SLMs

Less is More: Recursive Reasoning with Tiny Networks

Paper: https://arxiv.org/html/2510.04871v1

Abstract

Hierarchical Reasoning Model (HRM) is a novel approach using two small neural networks recursing at different frequencies. This biologically inspired method beats Large Language models (LLMs) on hard puzzle tasks such as Sudoku, Maze, and ARC-AGI while trained with small models (27M parameters) on small data (~ 1000 examples). HRM holds great promise for solving hard problems with small networks, but it is not yet well understood and may be suboptimal. We propose Tiny Recursive Model (TRM), a much simpler recursive reasoning approach that achieves significantly higher generalization than HRM, while using a single tiny network with only 2 layers. With only 7M parameters, TRM obtains 45% test-accuracy on ARC-AGI-1 and 8% on ARC-AGI-2, higher than most LLMs (e.g., Deepseek R1, o3-mini, Gemini 2.5 Pro) with less than 0.01% of the parameters.

SLMs

Mistral Small 3.2

Mistral Small is a 24B param LLM that

Running in Ollama

```
ollama pull hf.co/gabriellarson/Mistral-Small-3.2-24B-Instruct-2506-GGUF:Q4_K_M
```

References

- https://simonwillison.net/2025/Jun/20/mistral-small-32/

SLMs

Hierarchical Reasoning Model

Paper URL: https://arxiv.org/pdf/2506.21734

Code Repo: https://github.com/sapientinc/HRM

HRM is an alternative to transformer architecture that is better able to reason. It outperforms transformer-based LLMs at ARC-AGI2 with only 27M parameters.

Training a 27M Parameter Model with 1000 Examples

In the paper the authors refer to the fact that they only use between 1000 and 10,000 examples for specific problem domains:

- **Sudoku-Extreme**: 1000 training examples (used in main experiments)
- **Sudoku-Extreme-Full**: ~10,000 examples (used in analysis experiments for convergence guarantees)
- **ARC-AGI**: ~1000 examples from the official dataset, heavily augmented with translations, rotations, flips, and color permutations

This may seem quite low considering that this is a 27M parameter neural network and it seems likely that the network would be underfit after so few examples. The authors provide some additional clarifications around this point:

1. Data augmentation is used in order to functionally boost the size of the training set.
2. The authors use deep supervision to augment the training process (rather than relying on back-propagation alone).
3. The problem domain is simpler than for language - particularly for things like Sudoku and ARC-AGI - these are structured grid type problems.

Claude Code

Using Claude Code with LiteLLM

As per: https://docs.litellm.ai/docs/tutorials/claude_responses_api

export env vars corresponding to your litellm proxy instance:

```
export ANTHROPIC_BASE_URL="http://0.0.0.0:4000"
export ANTHROPIC_AUTH_TOKEN="$LITELLM_MASTER_KEY"
```

Using Claude Code with LM Studio

Reference

Start the server:

```
lms server start --port 1234
```

Point Claude Code at local instance:

```
export ANTHROPIC_BASE_URL=http://localhost:1234
export ANTHROPIC_AUTH_TOKEN=lmstudio
```

Github MCP

Public project here: https://github.com/github/github-mcp-server

Generate a PAT

Enable the tool in Claude Code

```
 claude mcp add --transport http github https://api.githubcopilot.com/mcp/ --header
"Authorization: Bearer <token>"
```

OpenCode

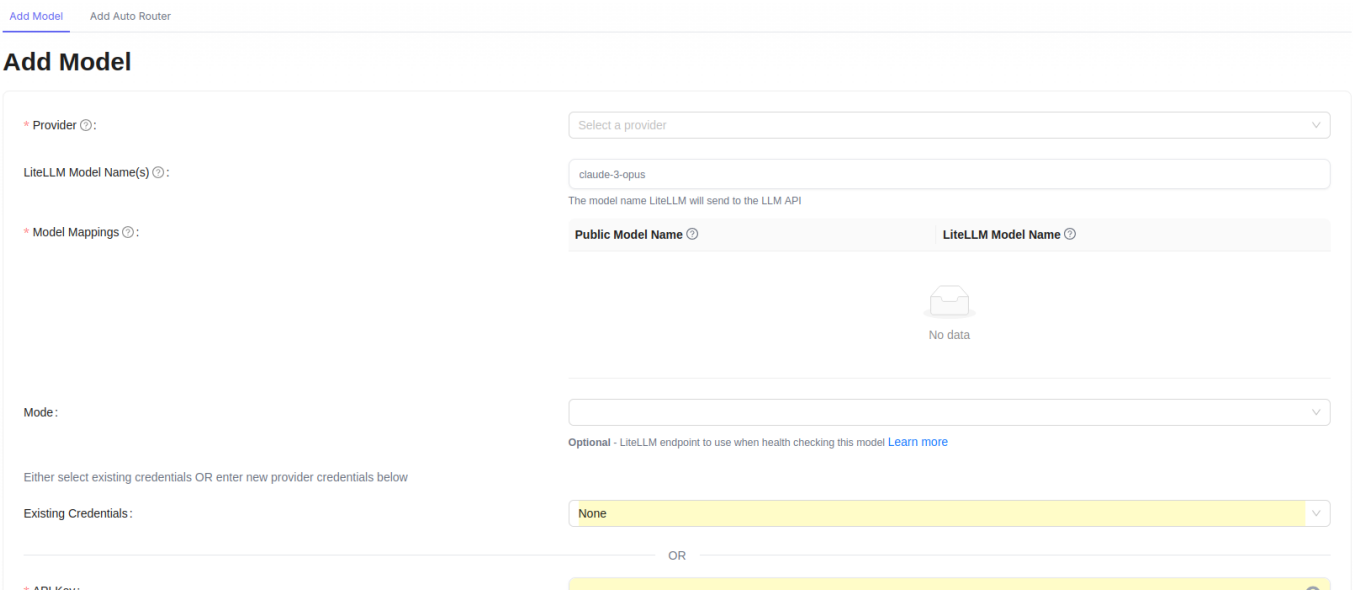
OpenCode is a tool for AI driven development. It allows you to set up a web-ui or a TUI and talk to an agent. You can also review outputs and directly run bash commands.

OpenCode with LiteLLM

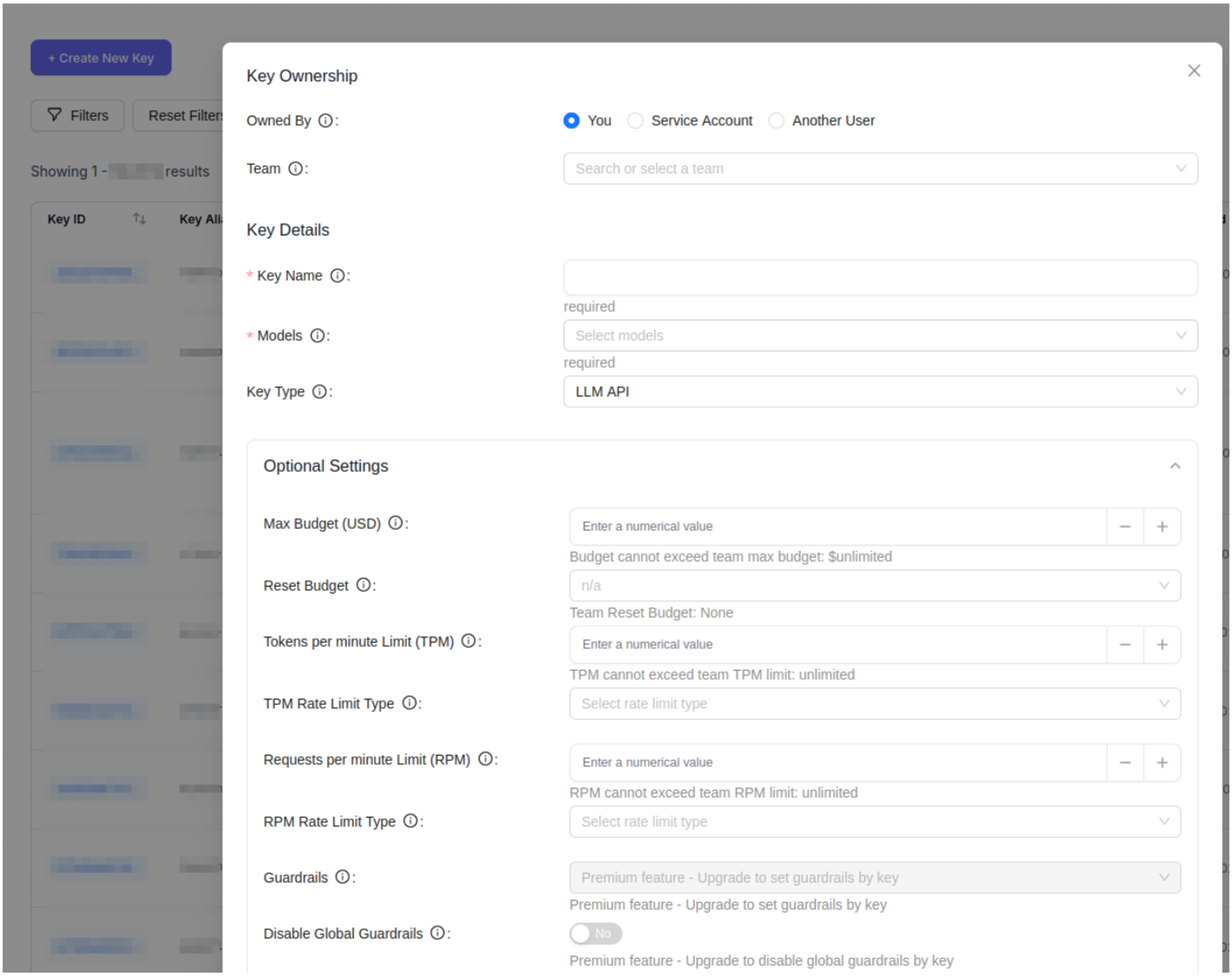
LiteLLM provides a way to connect your environment to multiple different models and model providers without giving OpenCode a bunch of different api keys. You can also closely track OpenCode-specific spending from one admin portal.

Configure LiteLLM

Firstly, add any models that you want to interact with, this might be via openrouter, OpenAI, Anthropic or local models. You'll want to keep track of the Public Model Name as that will be the name we pass on to OpenCode.

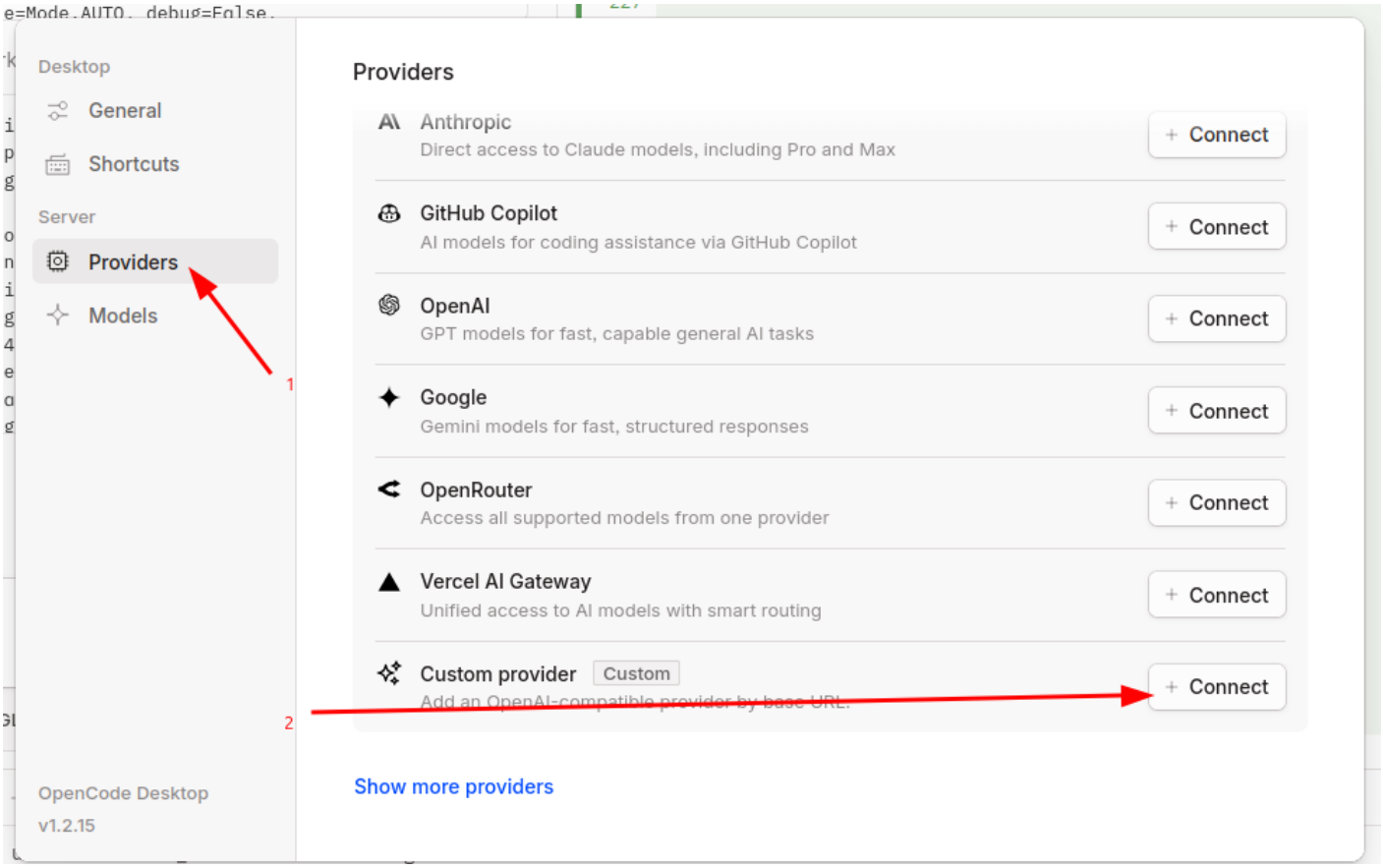


We'll add a new virtual API key so that we can provide this to OpenCode and track spending. You might also want to set a max budget which resets after N days.

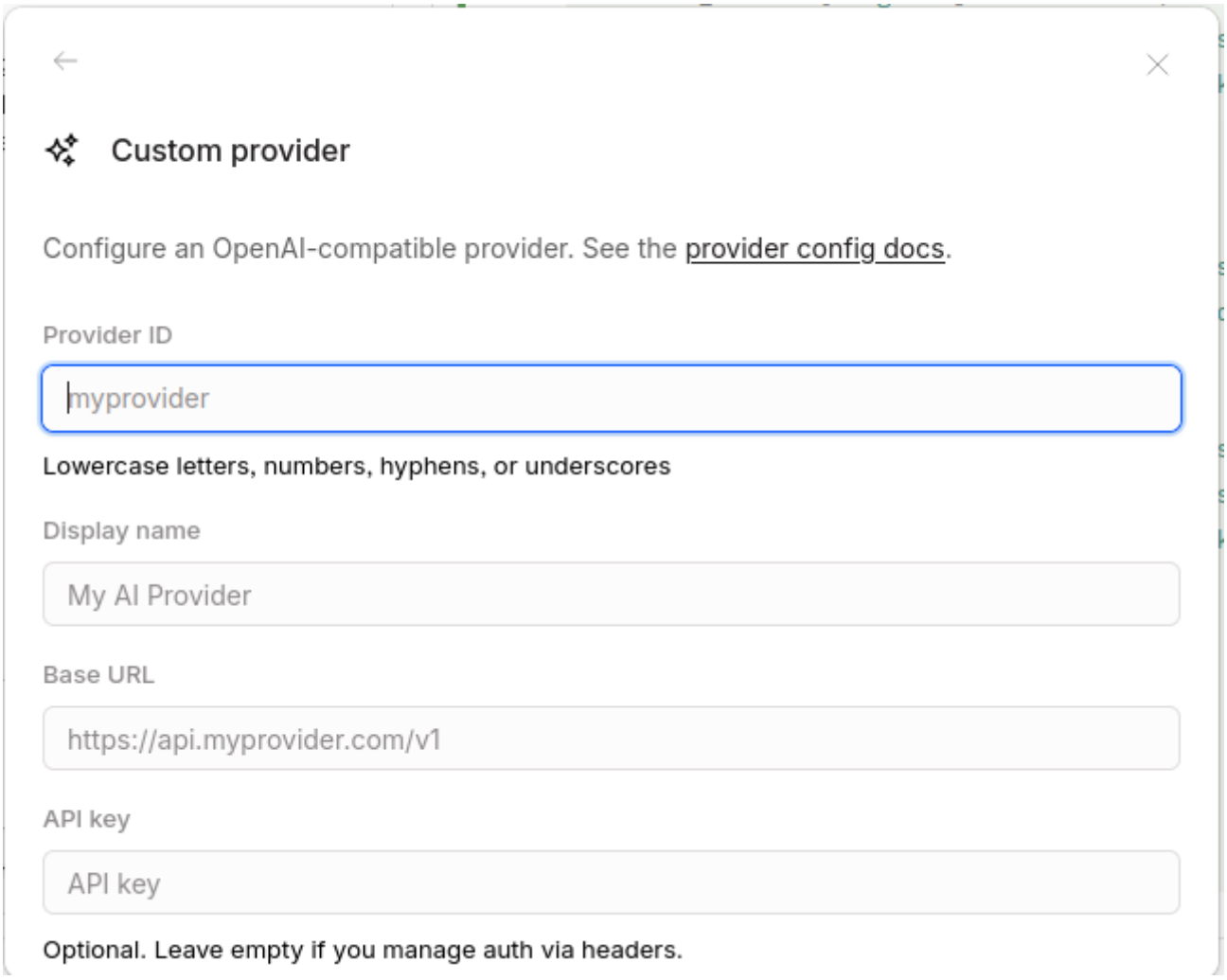


Configure OpenCode

Open the settings panel in the web UI and select "Providers" and click "Connect" next to custom provider.



Fill out the information from Litellm including the API base (remember to append /v1) and the API key you generated


Custom provider
Configure an OpenAI-compatible provider. See the provider config docs.
Provider ID
myprovider
Lowercase letters, numbers, hyphens, or underscores
Display name
My AI Provider
Base URL
https://api.myprovider.com/v1
API key
API key
Optional. Leave empty if you manage auth via headers.

Evaluating AGENTS.md: Are Repository-Level Context Files Helpful for Coding Agents?

Paper: 2602.11988

Authors: Thibaud Gloaguen, Niels Mündler, Mark Müller, Veselin Raychev, Martin Vechev

Published: February 2026

Summary

This paper evaluates whether repository-level context files (like) actually help coding agents perform better on real-world software engineering tasks.

Key Findings

Performance Impact

- **LLM-generated context files**: Decrease task success rates by ~3% on average
- **Developer-provided context files**: Marginally improve performance by ~4% on average
- **No context files**: Baseline performance

Cost Impact

- Context files increase inference costs by **over 20%** on average
- More steps required to complete tasks (2.45-3.92 additional steps)

Behavioral Changes

- **More testing and exploration**: Agents run more tests, search more files, read more files

- **Instruction following**: Agents generally follow instructions in context files
- **Redundant documentation**: Context files are often redundant with existing documentation
- **No effective overviews**: Context files don't provide useful repository overviews

AGENTBENCH

The authors created a new benchmark called **AGENTBENCH** consisting of:

- **138 unique instances** from 12 repositories
- Real GitHub issues (bug-fixing and feature addition tasks)
- Developer-written context files
- Python software engineering tasks

AGENTBENCH complements SWE-BENCH LITE (which uses popular repositories without context files).

Experimental Setup

Coding Agents Evaluated

- **CLAUDE CODE** with SONNET-4.5
- **CODEX** with GPT-5.2 and GPT-5.1 MINI
- **QWEN CODE** with QWEN3-30B-CODER

Datasets

- **SWE-BENCH LITE**: 300 tasks from 11 popular Python repositories (no context files)
- **AGENTBENCH**: 138 tasks from 12 repositories with developer-provided context files

Settings Evaluated

1. **NONE**: No context files
2. **LLM**: LLM-generated context files (using agent-developer recommendations)
3. **HUM**: Developer-provided context files

Key Insights

1. Context Files Make Tasks Harder

Instructions in context files increase reasoning tokens by 14-22%, suggesting tasks become more complex.

2. Context Files Are Redundant

When documentation files are removed from repositories, LLM-generated context files actually improve performance by 2.7% on average.

3. Stronger Models Don't Generate Better Context Files

Using GPT-5.2 to generate context files improves SWE-BENCH LITE performance by 2% but degrades AGENTBENCH performance by 3%.

4. Context Files Encourage Exploration

Agents use more repository-specific tools (e.g., ,) and run more tests when context files are present.

Recommendations

1. **Omit LLM-generated context files** for now, contrary to agent-developer recommendations
2. **Include only minimal requirements** in context files (e.g., specific tooling to use)
3. **Human-written context files** should describe only essential information
4. **Future work**: Improve automatic generation of concise, task-relevant guidance

Limitations

- Evaluation focused heavily on Python (a language well-represented in training data)
- Context files are a recent development (August 2025)
- Popular repositories used in benchmarks may not be representative of most codebases

Related Work

- **SWE-BENCH**: Repository-level coding agent evaluation
- **AGENTBENCH**: New benchmark for context file evaluation
- **Context files**: AGENTS.md, CLAUDE.md, README files for agents

Conclusion

Context files have only a **marginal effect** on coding agent behavior. While they encourage broader exploration and instruction following, they don't provide effective repository overviews and often make tasks harder. The authors recommend omitting LLM-generated context files and including only minimal requirements in human-written ones.

Tags: #agents #context-files #evaluation #SWE-bench #LLM-agents

Gemma 4

Gemma 4

Released **March 31, 2026** by Google DeepMind. Apache 2.0 licensed. Multimodal (text + image, audio on small models).

Model Sizes

Model	Type	Effective Params	Context	Modalities
E2B	Dense	2.3B (5.1B w/ embeddings)	128K	Text, Image, Audio
E4B	Dense	4.5B (8B w/ embeddings)	128K	Text, Image, Audio
26B A4B	MoE	3.8B active / 25.2B total	256K	Text, Image
31B	Dense	30.7B	256K	Text, Image

The **26B A4B** is the standout — a MoE model that runs almost as fast as a 4B model despite 26B total params. The **E2B/E4B** use Per-Layer Embeddings for on-device efficiency.

Local Running Options

1. **Ollama** — `ollama run google/gemma-4` (all sizes). Easiest one-command setup.
2. **llama.cpp** — GGUF quantized versions available on Hugging Face. Good for CPU/GPU hybrid inference.
3. **vLLM** — For higher-throughput server deployment. Supports the native HF safetensor weights.
4. **LM Studio** — GUI-based, supports GGUF formats. Good for desktop use.
5. **Hugging Face Transformers** — Direct Python API. Full precision or QLoRA fine-tuning.

Hardware Requirements (rough)

- **E2B (2.3B eff.)** — Runs on phones, any modern laptop (4-8 GB RAM)
- **E4B (4.5B eff.)** — 8-16 GB RAM, most 2024+ MacBooks

- **26B A4B** — 16-24 GB VRAM (single GPU), or CPU with enough RAM
- **31B** — 24-48 GB VRAM (A100/H100 recommended), or multi-GPU

The **26B A4B** is generally considered the sweet spot for local use — frontier-level benchmarks (82.6 MMLU Pro, 88.3 AIME) with ~4B active parameter compute cost.

All models are on Hugging Face under `google/gemma-4-*`.

AI Agent Resources

Resources, tools, and frameworks for AI coding agents and autonomous AI systems

AI Agent Resources

This page collects useful resources for effective **agentic engineering** — tools, frameworks, and skills that make coding agents more capable, reliable, and productive.

Each entry is something worth integrating into your agent workflow, whether as a skill, plugin, or standalone tool.

Impeccable

URL: impeccable.style · GitHub

Stars: 27k+

A design-focused AI agent skill that teaches your coding agent how design works. It loads seven reference files on every prompt (typography, color, motion, spatial, interaction, responsive, UX writing) to give your AI a shared design vocabulary.

Key features:

- **23+ design commands** — /typeset, /colorize, /animate, /layout, /critique, /audit, /polish, /distill, /adapt, and more
- **Design intelligence layer** — before any code runs, Impeccable establishes design context
- **Create → Evaluate → Refine → Simplify workflow** — structured design flow for coding agents
- **Codex asset producer agent** — v3.1.0 adds critique persistence and palette-first image flow
- **Cross-agent support** — works with multiple coding agent platforms

Use case: When your AI agent needs design-aware output — typography, color systems, spacing, motion, responsive layouts, and UX writing.

Ultrapowers

URL: github.com/danduma/ultrapowers

Stars: 1

A framework that adds a Product Manager role to your coding agent. It introduces PM-style thinking — requirements, prioritization, scope management — into the agent workflow.

Key features:

- **PM capabilities** — adds product management perspective to coding agents
- **Multi-agent support** — plugins for Claude, Codex, Cursor, and OpenCode
- **Structured agent hooks** — hooks, agents, commands, and scripts directories
- **Active development** — 454 commits, regularly updated

Use case: When you want your coding agent to think more like a PM — scoping work, managing priorities, and making product decisions before coding.

Playwright CLI

URL: github.com/microsoft/playwright-cli

Stars: 10k+

Microsoft's official CLI for common Playwright actions. Record and generate Playwright code, inspect selectors, and take screenshots — all from the command line.

Key features:

- **Record mode** — record browser interactions and generate Playwright code automatically
- **Code generation** — generate tests in multiple languages (TypeScript, Python, Java, C#)
- **Selector inspection** — inspect and test CSS selectors in the browser
- **Screenshot capture** — take screenshots and PDFs of web pages from the CLI
- **Multi-language support** — generated code for TypeScript, Python, Java, and C#

Use case: When you need to record browser interactions, generate Playwright test code quickly, or inspect selectors without opening the full Playwright Inspector.

Taste Skill

URL: tasteskill.dev · GitHub

A design-system skills pack for AI coding agents focused on producing premium, distinctive frontend output — the creator's pitch is "escape the generic AI slop." It ships as a collection of

SKILL.md files that plug into any agent supporting the skills protocol (Cursor, Claude Code, Codex, Gemini CLI, v0, Lovable, OpenCode, and more).

Key features:

- **Modular skill set** — separate skills for different goals: `taste-skill` (default all-rounder), `gpt-tasteskill` (stricter for GPT/Codex), `image-to-code-skill` (image-first workflow), `redesign-skill` (visual audit), `soft-skill` (calm/expensive aesthetic), `minimalist-skill` (editorial), `brutalist-skill` (Swiss typography, raw structure), and `stitch-skill` (Google Stitch-compatible export)
- **Image generation skills** — `imagegen-frontend-web`, `imagegen-frontend-mobile`, and `brandkit` for producing design reference images with strong art direction
- **Execution guard** — `output-skill` prevents placeholder code and skipped sections
- **Open beta** — v2 is in open beta with downloadable files

Use case: When you want your AI agent to produce frontend output with a cohesive, high-end visual identity rather than generic default styling. Install the skill pack and pick the aesthetic that matches your project.

Layers

URL: layers.jamiemill.com · GitHub

A design-thinking skills pack by Jamie Mill that gives AI agents a structured framework for product design decisions. Rather than jumping straight to screens, Layers walks the agent through seven layers of product design — from surface-level visuals down to user needs, domain knowledge, and observed behavior.

The Seven Layers:

1. **Surface** — visual design, colors, typography
2. **Interaction structure & flow** — navigation, states, edge cases
3. **Conceptual model** — objects, relationships, mental models
4. **Product & service strategy** — outcomes, bets, positioning
5. **User needs** — job stories, priorities, pain points
6. **The domain** — terminology, constraints, expertise
7. **Observed behaviour** — analytics, usage patterns, real-world data

Key features:

- **Diagnostic command** — `/layers-orient` audits all seven layers and tells you which is the bottleneck
- **Layer-specific skills** — individual `/layers-*` commands for deep dives (e.g., `/layers-user-needs`, `/layers-conceptual-model`, `/layers-interaction-flow`, `/layers-domain`)

- **Structured output** — produces markdown and mermaid diagrams (job stories, strategy trees, object maps, decision inventories)
- **MCP integrations** — can write decisions directly to Linear, Notion, Figma, or GitHub
- **Open source** — installable via `npx skills add jamiemill/layers-skills`

Use case: When you're starting a new feature or redesign and need your AI agent to think through the *right* design decisions before touching a pixel. Especially valuable when your team can't agree on direction — the framework surfaces what you're actually disagreeing about.

Superdesign

URL: app.superdesign.dev

An AI-powered design tool that generates production-ready website designs from text prompts, screenshots, or existing URLs. It combines a prompt library with AI generation and offers Cursor/Claude Code/VS Code integration for seamless handoff to development.

Key features:

- **Multi-modal input** — generate designs from text prompts, recreate from screenshots ("Recreate Screenshot"), or import from existing sites ("Import from Site")
- **Prompt library** — curated design templates including brutalist e-commerce, high-contrast landing pages, SaaS layouts, and editorial styles — each with detailed design system specifications (colors, typography, spacing, grid)
- **Design system integration** — use your own design system or let Superdesign generate one
- **AI model selection** — supports Gemini 3 Flash and other models for generation
- **Animation & effects** — explore and apply scroll-triggered animations, transitions, and visual effects
- **IDE integration** — works with Cursor, Claude Code, and VS Code for direct code generation

Use case: When you need rapid design exploration or want to generate a polished landing page / component library from a prompt. Great for early-stage projects, design sprints, or when you want to quickly validate a visual direction before committing to implementation.

Archify

URL: github.com/tt-a1i/archify

Stars: 523 · **Forks:** 33

A Claude Skill that generates polished architecture, technical workflow, sequence, data-flow, and lifecycle diagrams from plain-English descriptions. It outputs a single self-contained HTML file you can open in a browser, toggle between dark/light themes, copy to clipboard, or export at high resolution (4× PNG, JPEG, WebP, SVG).

Key features:

- **Natural language to diagrams** — describe your architecture or process in English, get a rendered diagram
- **Diagram types** — architecture diagrams, technical workflows, approvals, tool call chains, CI/CD pipelines, runbooks, request call chains, data pipelines, PII boundaries, and state machines
- **Theme toggle** — one-click switch between dark and light modes, persisted across sessions
- **Export options** — copy to clipboard (paste into Slack, Notion, GitHub) or export as 4× PNG, JPEG, WebP, or SVG
- **Single HTML output** — no dependencies, works offline, embeddable anywhere

Use case: When you need to quickly visualize system architecture, data flows, or technical processes and want a clean, exportable diagram without opening a drawing tool. Especially useful in the middle of coding sessions when you need to document or share a design.