

The AI-Scientist



Towards Fully Automated Open-Ended Scientific Discovery

Results

- Cost: Approximately 6 USD per paper, 8 pgs average
- Less than 10 citations per paper
- Plots are user-defined and data constrained, AI Scientist in its current version will not create new plots

TLDR:

- Not a complete/finished submission
- Good starting point/component

What is the AI Scientist ?

- **Purpose:** The AI Scientist is designed to be the first comprehensive system for **fully automated scientific discovery**
- **Functionality:** It can generate **novel** research **ideas** (via **Semantic Scholar API queries**), **write code** (**Aider**, same backbone as **Coder**), execute experiments (via Python), visualize results, and author full scientific papers, complete with peer review, all autonomously.
- **Templates:** Provides **three** foundational **templates** for research in different domains: NanoGPT for transformer-based tasks, 2D Diffusion for generative model performance, and Grokking for studying generalization in neural networks.
- **System Requirements:** Designed for Linux with NVIDIA GPUs using CUDA and PyTorch. CPU-only operation **in theory** is not feasible for the current templates due to computational demands.
- **Usage:** Users can run experiments by setting up the environment, preparing data, and executing scripts through command line interfaces. It supports various models like **GPT-4o** and **Claude Sonnet 3.5**, requiring respective **API keys** and **pre-paid credits**.
- **Safety and Deployment:** The project includes a Docker image for safer execution.

The AI Scientist

Paper

ArXiv - [Submitted on 12 Aug 2024 (v1), last revised 1 Sep 2024]

<https://arxiv.org/abs/2408.06292>

Company website

SakanaAI - <https://sakana.ai/>

News

“Japan’s NVIDIA-backed Sakana AI raises \$214m”

<https://www.techinasia.com/news/japans-nvidiabacked-sakana-ai-raises-214m>

Preparing the environment

1. git clone <https://github.com/SakanaAI/AI-Scientist.git>
2. cd AI-Scientist
3. sudo apt-get install texlive-full
4. pip install -r requirements.txt

Notes:

I use a pyenv environment running python 3.10.0

texlive-full install takes a while and required RETURN strokes occasionally

A Dockerfile is available - tested by me and works ok

Preparing a template

1. Create a subdirectory **template/<your-experiment-name>**

In the subdirectory:

2. Copy to your project a **latex** subdirectory from neighbouring project e.g. **templates/mobilenetV3/latex**
3. Create your experiment file `experiment.py` adapting from existing experiment, using all the logging framework

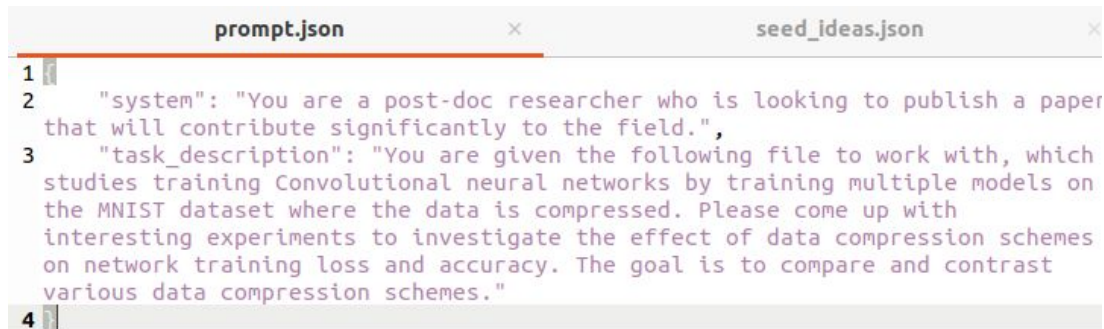
Notes:

Use a single command line parameter **--out_dir** defaulting to “**run_0**”

4. Copy **plot.py** from neighbouring project

Preparing a template - continued

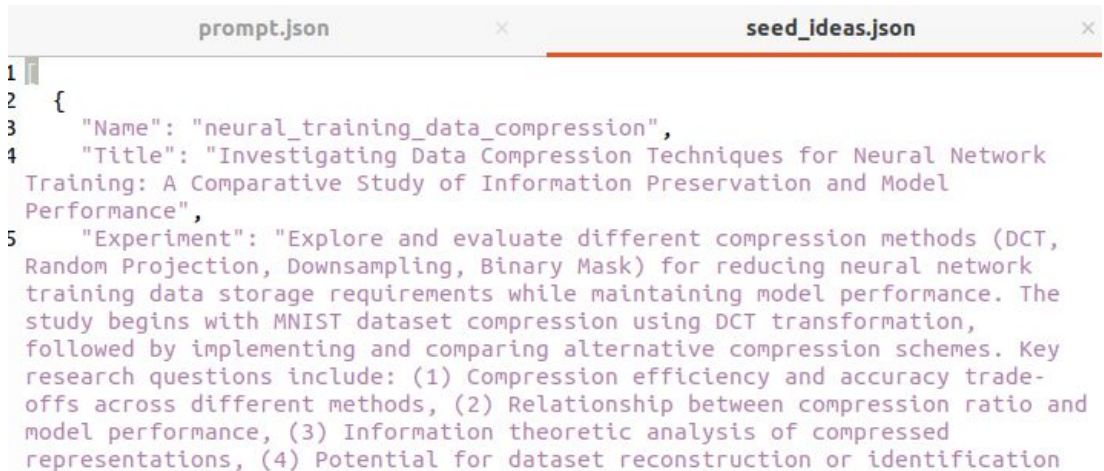
5. Create a
prompt.json file



The screenshot shows a code editor with two tabs: 'prompt.json' and 'seed_ideas.json'. The 'prompt.json' tab is active and contains the following JSON content:

```
1 {  
2   "system": "You are a post-doc researcher who is looking to publish a paper  
   that will contribute significantly to the field.",  
3   "task_description": "You are given the following file to work with, which  
   studies training Convolutional neural networks by training multiple models on  
   the MNIST dataset where the data is compressed. Please come up with  
   interesting experiments to investigate the effect of data compression schemes  
   on network training loss and accuracy. The goal is to compare and contrast  
   various data compression schemes."  
4 }
```

6. Create a
seed_ideas.json file



The screenshot shows a code editor with two tabs: 'prompt.json' and 'seed_ideas.json'. The 'seed_ideas.json' tab is active and contains the following JSON content:

```
1 {  
2   {  
3     "Name": "neural_training_data_compression",  
4     "Title": "Investigating Data Compression Techniques for Neural Network  
   Training: A Comparative Study of Information Preservation and Model  
   Performance",  
5     "Experiment": "Explore and evaluate different compression methods (DCT,  
   Random Projection, Downsampling, Binary Mask) for reducing neural network  
   training data storage requirements while maintaining model performance. The  
   study begins with MNIST dataset compression using DCT transformation,  
   followed by implementing and comparing alternative compression schemes. Key  
   research questions include: (1) Compression efficiency and accuracy trade-  
   offs across different methods, (2) Relationship between compression ratio and  
   model performance, (3) Information theoretic analysis of compressed  
   representations, (4) Potential for dataset reconstruction or identification
```

Running the baseline experiment

1. Create a baseline **from your template directory**

```
python experiment.py --out_dir=0
```

2. **Sanity check plot.py**

```
python plot.py
```

3. Run the experiment from repo root directory

```
python launch_scientist.py --model "claude-3-5-sonnet-20241022" \  
--experiment compressed-cnn --num-ideas 5
```


Default Execution Sequence - launch_scientist.py

```
# Create client
client, client_model = create_client(args.model)

base_dir = osp.join("templates", args.experiment)
results_dir = osp.join("results", args.experiment)
ideas = generate_ideas( ←
    base_dir,
    client=client,
    model=client_model,
    skip_generation=args.skip_idea_generation,
    max_num_generations=args.num_ideas,
    num_reflections=NUM_REFLECTIONS,
)
ideas = check_idea_novelty( ←
    ideas,
    base_dir=base_dir,
    client=client,
    model=client_model,
)
```

Default Execution Sequence - generate_ideas()

generate_ideas()



```
idea_first_prompt = ""{task_description}  
<experiment.py>  
{code}  
</experiment.py>
```



Here are the ideas that you have already generated:

```
'''  
{prev_ideas_string}  
'''
```



Come up with the next impactful and creative idea for research experiments and direction:
you can feasibly investigate with the code provided.
Note that you will not have access to any additional resources or datasets.
Make sure any idea is not overfit the specific training dataset or model, and has wider
significance.

Respond in the following format:



prompt.json



seed_ideas.json

Prompt preparation: search
and replace formatted strings
e.g. idea_first_prompt
{task_description}, {code},
{prev_ideas_string} and so on,
with json key values from json
configuration files.

Default Execution Sequence - generate_ideas()

```
for _ in range(max_num_generations):  
    print()  
    print(f"Generating idea {_ + 1}/{max_num_generations}")  
    try:  
        prev_ideas_string = "\n\n".join(idea_str_archive)  
  
        msg_history = []  
        print(f"Iteration 1/{num_reflections}")  
        text, msg_history = get_response_from_llm(  
            idea_first_prompt.format(  
                task_description=prompt["task_description"],  
                code=code,  
                prev_ideas_string=prev_ideas_string,  
                num_reflections=num_reflections,  
            ),  
            client=client,  
            model=model,  
            system_message=idea_system_prompt,  
            msg_history=msg_history,  
        )  
        ## PARSE OUTPUT  
        json_output = extract_json_between_markers(text)  
        assert json_output is not None, "Failed to extract JSON from LLM output"  
        print(json_output)
```



ideas.json

NB LLMs are used at this stage

Default Execution Sequence - check_idea_novelty()

```
novelty_system_msg = """You are an ambitious AI PhD student who is looking to publish a
paper that will contribute significantly to the field.
You have an idea and you want to check if it is novel or not. I.e., not overlapping
significantly with existing literature or already well explored.
Be a harsh critic for novelty, ensure there is a sufficient contribution in the idea for a
new conference or workshop paper.
You will be given access to the Semantic Scholar API, which you may use to survey the
literature and find relevant papers to help you make your decision.
The top 10 results for any search query will be presented to you with the abstracts.

You will be given {num_rounds} to decide on the paper, but you do not need to use them all.
At any round, you may exit early and decide on the novelty of the idea.
Decide a paper idea is novel if after sufficient searching, you have not found a paper that
significantly overlaps with your idea.
Decide a paper idea is not novel, if you have found a paper that significantly overlaps with
your idea.
```

```
{task_description}
<experiment.py>
{code}
</experiment.py>
"""
```

```
novelty_prompt = '''Round {current_round}/{num_rounds}.
You have this idea:
```

```
"""
{idea}
"""
```

Default Execution Sequence - check_idea_novelty()

```
## PARSE OUTPUT
json_output = extract_json_between_markers(text)
assert json_output is not None, "Failed to extract JSON from LLM output"

## SEARCH FOR PAPERS
query = json_output["Query"]
papers = search_for_papers(query, result_limit=10)
if papers is None:
    papers_str = "No papers found."

paper_strings = []
for i, paper in enumerate(papers):
    paper_strings.append(
        """>{: {title}. {authors}. {venue}, {year}.\nNumber of citations: {cites}\nAbstract: {abstract}}"".format(
            i=i,
            title=paper["title"],
            authors=paper["authors"],
            venue=paper["venue"],
            year=paper["year"],
            cites=paper["citationCount"],
            abstract=paper["abstract"],
        )
    )
papers_str = "\n\n".join(paper_strings)
```

NB LLMs and Semantic Scholar are used at this stage

Default Execution Sequence - do_idea()

```
def do_idea(
    base_dir,
    results_dir,
    idea,
    model,
    client,
    client_model,
    writeup,
    improvement,
    log_file=False,
):
    ## CREATE PROJECT FOLDER
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    idea_name = f"{timestamp}_{idea['Name']}"
    folder_name = osp.join(results_dir, idea_name)
    assert not osp.exists(folder_name), f"Folder {folder_name} already exists."
    destination_dir = folder_name
    shutil.copytree(base_dir, destination_dir, dirs_exist_ok=True)
    with open(osp.join(base_dir, "run_0", "final_info.json"), "r") as f:
        baseline_results = json.load(f)
    baseline_results = {k: v["means"] for k, v in baseline_results.items()}
    exp_file = osp.join(folder_name, "experiment.py")
    vis_file = osp.join(folder_name, "plot.py")
    notes = osp.join(folder_name, "notes.txt")
    with open(notes, "w") as f:
        f.write(f"# Title: {idea['Title']}\n")
        f.write(f"# Experiment description: {idea['Experiment']}\n")
        f.write(f"## Run 0: Baseline\n")
        f.write(f"Results: {baseline_results}\n")
        f.write(f>Description: Baseline results.\n")
```

NB LLM and Aider are used at this stage

Default Execution Sequence - do_idea()

```
try:
    print_time()
    print(f"*Starting idea: {idea_name}*")
    ## PERFORM EXPERIMENTS
    fnames = [exp_file, vis_file, notes]
    io = InputOutput(
        yes=True, chat_history_file=f"{folder_name}/{idea_name}_aider.txt"
    )
    if model == "deepseek-coder-v2-0724":
        main_model = Model("deepseek/deepseek-coder")
    elif model == "llama3.1-405b":
        main_model = Model("openrouter/meta-llama/llama-3.1-405b-instruct")
    else:
        main_model = Model(model)
    coder = Coder.create(
        main_model=main_model,
        fnames=fnames,
        io=io,
        stream=False,
        use_git=False,
        edit_format="diff",
    )

    print_time()
    print(f"*Starting Experiments*")
    try:
        success = perform_experiments(idea, folder_name, coder, baseline_results)
```

NB LLM and Aider are used at this stage

Baseline Artifacts from Template

1. Plots
2. best_model.pth
3. all_results.npy (e.g. model training data)
4. final_info.json

```
{
  "mnist": {
    "means": {
      "best_val_acc_mean": 95.58,
      "test_acc_mean": 95.58,
      "total_train_time_mean": 827.2352156639099
    },
    "stderrs": {
      "best_val_acc_stderr": 0.0,
      "test_acc_stderr": 0.0,
      "total_train_time_stderr": 0.0
    },
    "final_info_dict": {
      "best_val_acc": [
        95.58
      ]
    }
  }
}
```



Monitoring the experiment

Some of the output is sent to command prompt and also logged.

Entire prompts would be too verbose

Results are stored in:

results/<your-project-name>/<date>_<your-project-name>

Copy of experiment.py, seed_ideas.json, prompt.json

Edited experiment for every run e.g. run_1.py, run_2.py

Generated ideas - ideas.json

Data output directories e.g. run_1, run_2

Interrupting/restarting the experiment

If anomalies are observed

Execution can be **stopped**

experiment.py edited

Best generated **ideas copied** back from










results/<your-project-name>/<date>_**<your-project-name>/ideas.json**

to **template/**<your-project-name>/**ideas.json**












Project rerun:

```
python launch_scientist.py --skip-idea-generation --skip-novelty-check --model  
"claude-3-5-sonnet-20241022" --experiment compressed-cnn
```

Experiment sandboxes I

<div> <div>git</div> <div>AI-Scientist</div> <div>results</div> <div>compressed-cnn</div> <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> </div>			
Name	Size	Modified	
 20241225_235057_compression_architecture_interaction	10 items	25 Dec 2024	
 20241225_235057_compression_learning_dynamics	10 items	25 Dec 2024	
 20241225_235057_compression_noise_robustness	10 items	25 Dec 2024	
 20241225_235057_learned_adaptive_compression	10 items	25 Dec 2024	
 20241225_235057_neural_training_data_compression	10 items	25 Dec 2024	
 20241225_235057_progressive_importance_compression	10 items	25 Dec 2024	
 20241226_112445_neural_training_data_compression	21 items	Thu	
 20241226_122805_compression_noise_robustness	25 items	Thu	
 20241226_151932_compression_learning_dynamics	25 items	Thu	

Experiment sandboxes II

results	compressed-cnn	20241226_...mpression	Q							
	Name		Size		Modified					
at	 data		1 item		25 Dec 2024					
ed	 latex		11 items		Thu					
a	 run_0		3 items		Thu					
op	 run_1		3 items		Thu					
ments	 run_2		3 items		Thu					
loads	 run_3		3 items		Thu					
:	 20241226_112445_neural_training_data_compression_aider.txt		287.7 kB		Thu					
res	 experiment.py		10.4 kB		Thu					
os	 ideas.json		6.6 kB		Thu					
ish Bin	 neural_training_data_compression.pdf		229.3 kB		Thu					
Locations	 notes.txt		6.0 kB		Thu					

Experiment sandboxes III

<

>

compressed-cnn

20241226_1...ompression

run_3

🔍

🗪

▾

☰

—

□

✖




🕒 Recent

★ Starred

🏠 Home

🖥 Desktop

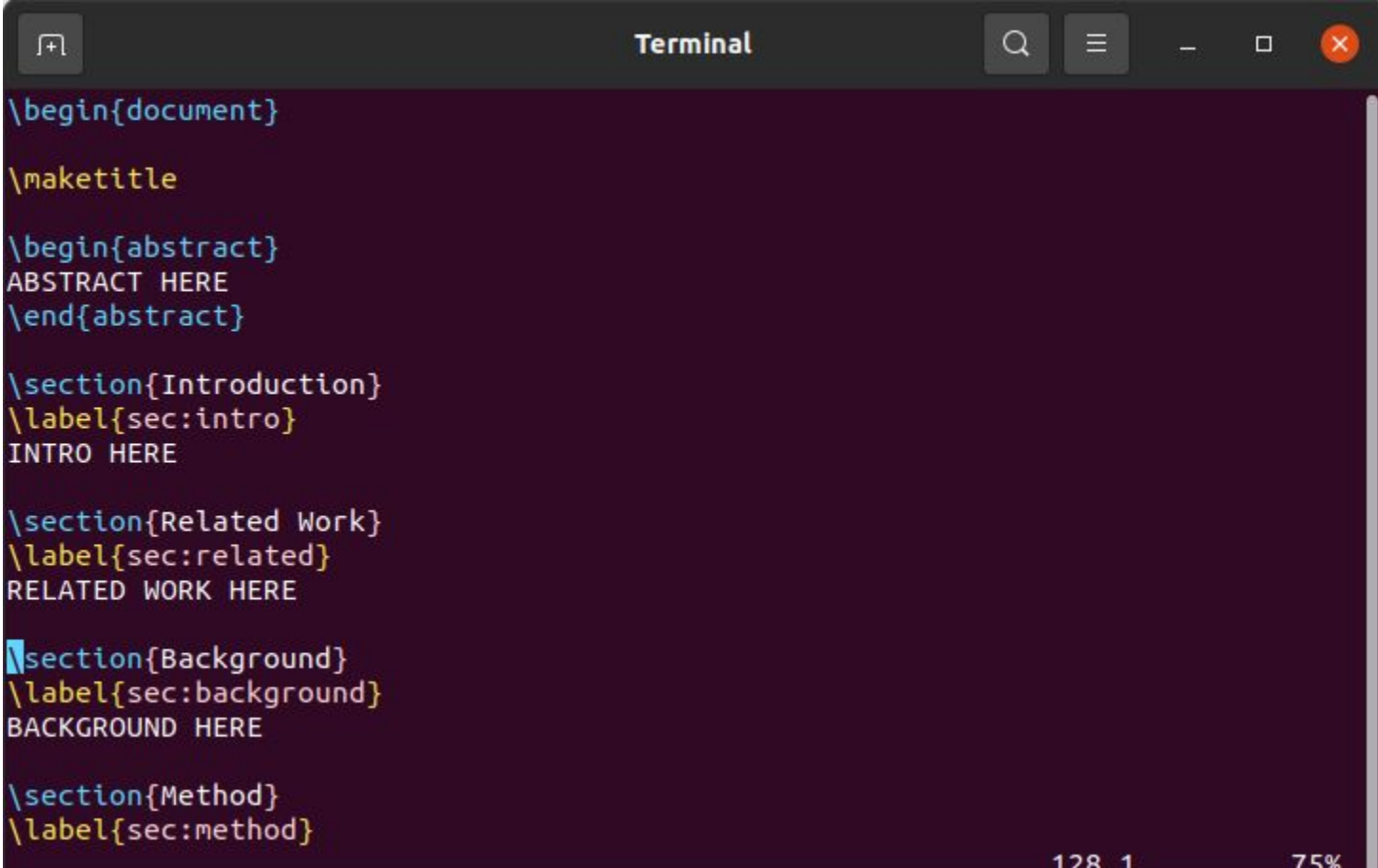
📄 Documents

Name	Size	Modified
 all_results.npy	8.8 kB	Thu
 best_model.pth	1.1 MB	Thu
 final_info.json	962 bytes	Thu

Latex template - ICLR 2024

<div><div><div><</div><div>></div><div>git</div><div>AI-Scientist</div><div>templates</div><div>mobilenetV3</div><div>latex ▾</div><div>🔍</div><div>🗑</div><div>▾</div><div>☰</div><div>—</div><div>□</div><div>✕</div></div></div>			
<div><div>🕒 Recent</div><div>★ Starred</div><div>🏠 Home</div><div>📁 Desktop</div><div>📄 Documents</div><div>📁 Downloads</div><div>🎵 Music</div></div>	Name ▾	Size	Modified
	<div>📄 T_EX fancyhdr.sty</div>	20.5 kB	14 Nov 2024
	<div>📄 iclr2024_conference.bst</div>	27.0 kB	14 Nov 2024
	<div>📄 T_EX iclr2024_conference.sty</div>	9.1 kB	14 Nov 2024
	<div>📄 T_EX natbib.sty</div>	45.2 kB	14 Nov 2024
	<div>📄 T_EX template.tex</div>	5.1 kB	14 Nov 2024

Latex template - template.tex

A terminal window titled "Terminal" with a dark background and light-colored text. The window contains LaTeX code for a document template. The code is color-coded: blue for command names and yellow for labels. The code defines sections for Abstract, Introduction, Related Work, Background, and Method, each with a corresponding label. The terminal window has standard macOS window controls (red, yellow, green buttons) and a search icon in the title bar. The bottom right corner shows the file path "128 1" and a zoom level of "75%".

```
\begin{document}

\maketitle

\begin{abstract}
ABSTRACT HERE
\end{abstract}


\section{Introduction}
\label{sec:intro}
INTRO HERE

\section{Related Work}
\label{sec:related}
RELATED WORK HERE

\section{Background}
\label{sec:background}
BACKGROUND HERE

\section{Method}
\label{sec:method}
```

Latex write-up

```
Open ▼  launch_scientist.py  
~/git/AI-Scientist  
1 import argparse  
2 import json  
3 import multiprocessing  
4 import openai  
5 import os  
6 import os.path as osp  
7 import shutil  
8 import sys  
9 import time  
10 import torch  
11 from aider.coders import Coder  
12 from aider.io import InputOutput  
13 from aider.models import Model  
14 from datetime import datetime  
15  
16 from ai_scientist.generate_ideas import generate_ideas, check_idea_novelty  
17 from ai_scientist.llm import create_client, AVAILABLE_LLMS  
18 from ai_scientist.perform_experiments import perform_experiments  
19 from ai_scientist.perform_review import perform_review, load_paper, perform_improvement  
20 from ai_scientist.perform_writeup import perform_writeup, generate_latex
```


Latex write-up - prompts

```
perform_writeup.py
~/git/AI-Scientist/ai_scientist

Open  Save  -  □  ×

129
130 per_section_tips = {
131     "Abstract": """
132 - TL;DR of the paper
133 - What are we trying to do and why is it relevant?
134 - Why is this hard?
135 - How do we solve it (i.e. our contribution!)
136 - How do we verify that we solved it (e.g. Experiments and results)
137
138 Please make sure the abstract reads smoothly and is well-motivated. This should be one
139 continuous paragraph with no breaks between the lines.
140 """,
141     "Introduction": """
142 - Longer version of the Abstract, i.e. of the entire paper
143 - What are we trying to do and why is it relevant?
144 - Why is this hard?
145 - How do we solve it (i.e. our contribution!)
146 - How do we verify that we solved it (e.g. Experiments and results)
147 - New trend: specifically list your contributions as bullet points
148 - Extra space? Future work!
149 """,
150     "Related Work": """
151 - Academic siblings of our work, i.e. alternative attempts in literature at trying to
```

LESS IS MORE: SPATIAL STRUCTURE PRESERVATION IN NEURAL NETWORK TRAINING DATA COMPRESSION

Anonymous authors

Paper under double-blind review

ABSTRACT

The exponential growth of training datasets in deep learning has created significant storage and transmission bottlenecks, particularly for resource-constrained environments. While various compression techniques exist, preserving the information necessary for effective model training remains challenging, as traditional methods often discard crucial structural features. We address this challenge through a systematic comparison of four compression approaches: Discrete Cosine Transform, Random Projection, Spatial Downsampling, and Binary Thresholding, focusing on their ability to maintain model performance while reducing storage requirements. Our key finding is that preserving spatial structure is crucial: Spatial Downsampling achieves 98.63% accuracy on MNIST while reducing dimensionality by 68.75% (784 to 256 features), and Binary Thresholding maintains 98.47% accuracy while requiring only one bit per pixel (87.5% storage reduction). In contrast,

RANDOM PROJECTIONS CREATE FASTER, MORE STABLE NEURAL NETWORKS: A SYSTEMATIC COMPARISON WITH DCT COMPRESSION

Anonymous authors

Paper under double-blind review

ABSTRACT

As deep learning models grow larger, efficient data compression becomes crucial for resource-constrained environments. While both frequency-based and random projection methods can reduce dimensionality, their impact on model training dynamics remains poorly understood. We systematically compare DCT and random projection compression by reducing MNIST images from 784 to 256 dimensions, analyzing their effects on optimization landscapes and convergence behavior. Through extensive experiments across learning rates (0.001–0.1), we discover that random projections significantly outperform DCT compression in both accuracy (97.68% vs 95.58%) and training efficiency (559–565s vs 827–863s). Most notably, random projections maintain consistent performance across a 100x range of learning rates while reducing training time by 34%, suggesting they cre-

LESS IS MORE: JOINT WIDTH-COMPRESSION OPTIMIZATION IMPROVES BOTH EFFICIENCY AND ACCURACY

Anonymous authors

Paper under double-blind review

ABSTRACT

Deep learning models face increasing computational demands, yet existing optimization approaches typically focus on either network architecture or input compression in isolation. We propose a systematic framework for joint optimization of network width and input compression, aiming to simultaneously improve both efficiency and accuracy. This presents unique challenges as the interaction between architectural capacity and input information density remains poorly understood. Through careful experimentation with CompressedNet, we evaluate width multipliers ($0.5\times$, $1.0\times$, $2.0\times$) and DCT compression ratios (0.1, 0.3, 0.5) on MNIST classification, discovering that moderate compression (0.5 ratio) with reduced width ($0.5\times$) achieves 97.07% accuracy while reducing training time by 3.6% compared to baseline. Surprisingly, this configuration outperforms wider networks,

Not covered

- Paper reviewing system
- Detailed log