Identifying and Mitigating Shortcut Learning via Task Arithmetic

Implementation Blueprint + Reproducible Codebase

February 23, 2026

Abstract

Shortcut learning causes high in-distribution performance but poor robustness under distribution shift. This document provides a complete implementation-ready protocol using task arithmetic and trajectory analysis. We define modular training, task-vector extraction, and post-hoc editing workflows; include concrete CLIs and reproducibility assets; and outline evaluation on vision and language shortcut benchmarks. The accompanying codebase in `shortcut_patcher/` is executable end-to-end.

1 Introduction

Neural models frequently exploit spurious correlations such as backgrounds, textures, or demographic proxies. These "shortcut" features are predictive in training data but fail under shifts [1, 3]. Task arithmetic offers a practical post-hoc intervention: for pretrained weights w_{pre} and task-finetuned weights w_T, define $v_T = w_T - w_{\text{pre}}$; adding/subtracting v_T edits behavior without re-training [2].

2 Method Overview

2.1 Task vectors and edits

Given a task vector v_T, edited parameters are:

$$w_{\text{edit}} = w_{\text{base}} + \alpha v_T, \tag{1}$$

where $\alpha = -1$ forgets task T and positive α amplifies/combines skills.

2.2 Trajectory diagnostics

We log checkpoints every k steps and analyze:

- PCA of flattened weights.
- Gradient alignment with final task vector.
- Linear probes for shortcut attributes.
- CCA across representations pre/post edit.

3 Experimental Protocol

Vision tasks: Cars, DTD, EuroSAT, GTSRB, MNIST, RESISC45, SUN397, SVHN, Waterbirds, CelebA.
Language tasks: GPT-2 toxicity fine-tuning with WikiText control.
Training defaults: AdamW, lr 1e−5, wd=0.1, cosine schedule, snapshots every 100 steps.

4 Repository Layout

```
shortcut_patcher/
  data/{vision_tasks.py,text_tasks.py}
  src/{train.py,task_vector.py,edit_model.py,analyze.py,utils.py,visualize
     .py}
  env/{Dockerfile,environment.yml}
  experiments/{logs,results,figures}
  run_pipeline.sh
  paper/{main.tex,references.bib}
```

5 Implementation Notes

5.1 Training and snapshot logging

```
python src/train.py --task WaterbirdsShortcut --model synthetic-mlp \
  --max-steps 2000 --snapshot-every 100 --output experiments/logs/
     waterbirds
```

5.2 Task-vector computation

```
python src/task_vector.py \
  --pretrained experiments/logs/waterbirds/pretrained.pt \
  --finetuned experiments/logs/waterbirds/final.pt \
  --output experiments/results/waterbirds_vector.pt
```

5.3 Model editing

```
python src/edit_model.py --model-ckpt experiments/logs/waterbirds/
   pretrained.pt \
  --task-vector experiments/results/waterbirds_vector.pt \
  --alpha -1.0 --output experiments/results/waterbirds_forget.pt
```

5.4 Trajectory analysis

```
python src/analyze.py --method pca \
  --trajectory experiments/logs/waterbirds/snapshots \
  --output experiments/results/pca_summary.txt
```

6 Evaluation

Report:

- In-distribution accuracy (target/control)
- OOD and worst-group accuracy
- Robustness gap
- Toxicity and perplexity for GPT-2
- Compute budget and wall-clock

Table 1: Vision Forgetting Benchmark Template

Method	Target Acc	Control Acc	Worst-Group Acc
Fine-tuned baseline	–	–	–
Gradient ascent unlearn	–	–	–
Random vector edit	–	–	–
Negative task vector	–	–	–

7 Mermaid Pipeline

```
flowchart LR
  A[Pretrained Model] --> B[Fine-tune on Task T]
  B --> C[Compute v_T = w_T - w_pre]
  C --> D[Apply alpha=-1 to forget]
  C --> E[Apply alpha=+1 for add/composition]
  D --> F[Evaluate ID/OOD/Worst-group]
  E --> F
```

8 Conclusion

This artifact provides a full, reproducible implementation for identifying shortcut directions and mitigating them with task arithmetic. The code is intentionally modular so real CLIP/GPT-2 pipelines can replace synthetic loaders with minimal changes.

References

[1] Robert Geirhos, J"orn-Henrik Jacobsen, Claudio Michaelis, et al. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

[2] Gabriel Ilharco, Marco Ribeiro, et al. Editing models with task arithmetic. In *ICLR*, 2023.

[3] Aditya Murali et al. Spurious features are unstable: Investigating shortcut dynamics at training time. In *NeurIPS Workshop*, 2023.