Contextual Head Gating (CHG) Implementation

A complete, production-ready implementation of Contextual Head Gating for analyzing attention head importance in transformer models.

Overview

CHG learns soft gating parameters for attention heads to identify:

- **Critical heads** (G+): Heads essential for model performance
- **Removable heads** (G-): Heads that can be pruned without hurting performance
- **Contrastive circuits**: Heads responsible for specific behaviors or biases

Installation

Requirements

bash

```bash
# Create environment
conda create -n chg python=3.10
conda activate chg

# Install dependencies
pip install torch transformers accelerate datasets wandb
pip install numpy pandas matplotlib seaborn tqdm
```

Repository Structure

```
chg-project/
├── README.md
├── requirements.txt
├── src/
│   ├── model/
│   │   └── gate_wrapper.py      # Core gating logic
│   ├── train/
│   │   ├── fit_gates.py         # Fit gate parameters
│   │   ├── contrastive_chg.py   # Contrastive CHG
│   │   └── eval_ablation.py     # Ablation experiments
│   └── analysis/
│       └── analyze.py           # Analysis & visualization
└── notebooks/
    └── quick_demo.ipynb         # Interactive demo
```

Quick Start

1. Fit Initialization Mask (λ=0)

This creates a baseline that preserves model behavior:

bash

```bash
python src/train/fit_gates.py \
  --model gpt2 \
  --dataset wikitext \
  --dataset_config wikitext-2-raw-v1 \
  --out gates_init.pt \
  --epochs 3 \
  --lr 0.01 \
  --lambda_reg 0.0 \
  --batch_size 4
```

2. Fit Retention Mask (λ<0, G+)

Identifies critical heads:

bash

```bash
python src/train/fit_gates.py \
  --model gpt2 \
  --dataset wikitext \
  --dataset_config wikitext-2-raw-v1 \
  --init gates_init.pt \
  --out gates_retention.pt \
  --epochs 3 \
  --lr 0.01 \
  --lambda_reg -0.1 \
  --batch_size 4
```

3. Fit Removal Mask (λ>0, G-)

Identifies removable heads:

bash

```bash
python src/train/fit_gates.py \
  --model gpt2 \
  --dataset wikitext \
  --dataset_config wikitext-2-raw-v1 \
  --init gates_init.pt \
  --out gates_removal.pt \
  --epochs 3 \
  --lr 0.01 \
  --lambda_reg 0.1 \
  --batch_size 4
```

4. Evaluate Head Importance

Compute individual head effects:

bash

```bash
python src/train/eval_ablation.py \
  --model gpt2 \
  --gates gates_init.pt \
  --mode individual \
  --output ablation_results.json
```

Or run sequential ablation:

bash

```bash
python src/train/eval_ablation.py \
  --model gpt2 \
  --gates gates_init.pt \
  --mode sequential \
  --ablation_order forward \
  --output sequential_ablation.json
```

5. Generate Analysis

Create visualizations and reports:

bash

```bash
python src/analysis/analyze.py \
  --results ablation_results.json \
  --gates gates_init.pt \
  --output_dir analysis_output \
  --top_k 50
```

This generates:

- `importance_heatmap.png`: Per-head importance scores
- `layer_importance.png`: Average importance by layer
- `gate_distribution.png`: Distribution of gate values
- `head_rankings.csv`: Ranked list of heads
- `summary_report.txt`: Text summary

Advanced Usage

Contrastive CHG

Learn gates that retain one dataset while forgetting another:

bash

```bash
python src/train/contrastive_chg.py \
  --model gpt2 \
  --retain_dataset wikitext \
  --forget_dataset <biased_dataset> \
  --out gates_contrastive.pt \
  --alpha 1.0 \
  --epochs 5
```

Custom Evaluation Prompts

Create a file `prompts.txt` with one prompt per line:

bash

```bash
python src/train/eval_ablation.py \
  --model gpt2 \
  --gates gates_init.pt \
  --prompts_file prompts.txt \
  --mode individual \
  --output custom_results.json
```

Multi-Seed Experiments

For robustness, run with multiple seeds:

bash

```
for seed in 42 123 456; do
  python src/train/fit_gates.py \
    --model gpt2 \
    --out gates_init_seed${seed}.pt \
    --seed $seed
done
```

Model Support

The implementation auto-detects model architectures and supports:

- **GPT-2** variants
- **LLaMA** family (LLaMA, LLaMA-2, LLaMA-3)
- **Mistral** models
- **OPT** models

For other models, the hook installation in `gate_wrapper.py` may need adaptation.

Key Parameters

Training Parameters

- `--lambda_reg`: Regularization strength
  - `0.0`: Initialization (preserve behavior)
  - `< 0`: Retention mask (encourage high gates)
  - `> 0`: Removal mask (encourage low gates)
  - Typical values: -0.1 to -0.5 (retention), 0.1 to 0.5 (removal)
- `--lr`: Learning rate (default: 0.01)
  - Gates typically need higher LR than standard fine-tuning
  - Range: 5e-3 to 5e-2
- `--epochs`: Training epochs
  - 3-5 for small datasets
  - 1-2 for large datasets

Evaluation Parameters

- `--mode`: Ablation mode
  - `individual`: Test each head independently (slower, more informative)
  - `sequential`: Ablate heads one by one (faster, shows cumulative effect)
- `--ablation_order`: For sequential mode
  - `forward`: Layers 0→N, heads 0→H
  - `backward`: Layers N→0, heads H→0
  - `random`: Random order

Understanding Results

Importance Scores

- **Negative scores**: Important heads (performance drops when ablated)

- **Positive scores**: Anti-important heads (performance improves when ablated)
- **Near-zero scores**: Neutral/redundant heads

Gate Values

After fitting:

- **High gates (>0.7)**: Critical heads to retain
- **Low gates (<0.3)**: Safe to prune
- **Mid gates (0.3-0.7)**: Context-dependent importance

Heatmap Interpretation

The heatmap shows importance by layer and head position:

- Early layers often handle low-level features
- Middle layers capture semantic patterns
- Late layers refine predictions

Common Issues & Solutions

1. Out of Memory

bash

```bash
# Reduce batch size and use gradient accumulation
python src/train/fit_gates.py \
  --batch_size 1 \
  --grad_accum_steps 4
```

2. Unstable Training

bash

```bash
# Lower learning rate
python src/train/fit_gates.py \
  --lr 0.005
```

3. Model Architecture Not Recognized

Check the model structure:

python

```python
from transformers import AutoModel
model = AutoModel.from_pretrained("model_name")
print(model)
```

Then adapt `get_attention_module()` in `gate_wrapper.py`.

4. Gates Not Changing

- Increase `--lambda_reg` magnitude
- Increase `--epochs`
- Check that hooks are installed correctly

Reproducibility

Set seeds for reproducible results:

bash

```bash
python src/train/fit_gates.py \
  --seed 42 \
  --dataset_split "train[:100]"  # Use fixed subset
```

Citation

If you use this implementation, please cite the CHG paper:

bibtex

```bibtex
@article{chg2024,
  title={Contextual Head Gating for Neural Network Interpretability},
  author={[Authors]},
  journal={[Venue]},
  year={2024}
}
```

Development Workflow

Day 1 checklist:

1. ✅ Clone repo and install dependencies
2. ✅ Run smoke test: `python src/train/fit_gates.py --model gpt2 --epochs 1`
3. ✅ Fit G0 (λ=0) initialization
4. ✅ Fit G+ and G- using saved initialization
5. ✅ Run ablation experiments
6. ✅ Generate visualizations

Tips for Research

1. **Always save initialization**: Use it as the starting point for both G+ and G-
2. **Multiple seeds**: Run 3-5 seeds and average results
3. **Dataset size**: Start small (1%), scale up gradually
4. **Validation**: Use held-out prompts for ablation evaluation
5. **Layer-wise analysis**: Check if patterns differ by layer depth

Performance Notes

- **Gate fitting**: ~2x slower than normal forward pass
- **Individual ablation**: $O(L \times H)$ forward passes (can be slow)
- **Sequential ablation**: $O(L \times H)$ forward passes but evaluates cumulative effect
- **GPU memory**: Same as model inference (weights frozen)

License

Support

For issues, questions, or contributions, please open an issue on GitHub.