OCTA Research – 365 Days of Learning, Theorizing, and Building AGI

Practical Fisher Geometry and RLCT Probes for Transformers: Estimation, Diagnostics, and Scaling Hooks

Week 1 · Day 4 · January 4, 2026

OCTA Research Internal Theory Program

Version 1.2b – Living Document Series (Refined and Fixed)

OCTA Research 365 Program Note. Day 1 disentangled macro "singularity" narratives from micro-level singularities in model architectures. Day 2 built a stratified geometric view of attention: tie manifolds, block cells, and parameter varieties. Day 3 introduced *Singular Learning Theory* (SLT) for Transformers, with the Real Log Canonical Threshold (RLCT) λ as effective dimension.
Day 4 moves from theory to instrumentation: we define practical estimators for Fisher information, Gauss–Newton matrices, and RLCT-like effective dimension in large-scale Transformers. We specify diagnostics and protocols that can be embedded into OCTA training pipelines, and design visual dashboards for *singular geometry in practice.*
Version 1.2 refines Day 4 by: (i) sharpening curvature regime definitions and RLCT links; (ii) adding a worked toy example with explicit Fisher and spectra; (iii) introducing OCTA Principles that tie curvature patterns to training and safety behavior; (iv) extending pseudo-code and engineering guidance; (v) polishing notation and visual schematics for deployment in the OCTA ecosystem.

Abstract

Deep models, including Transformers, are singular statistical models: their KL minimizer sets form manifolds or varieties, and their Fisher information matrices have degenerate spectra. Day 3 explained how the RLCT λ replaces parameter count in determining asymptotic generalization and Bayesian evidence. To exploit this in real systems, we need practical estimators and diagnostics.

This Day 4 document:

- distinguishes population Fisher, empirical Fisher, Gauss–Newton, and Hessian;
- describes scalable approximations (minibatch, Hutchinson trace, low-rank eigen-slices, layerwise/headwise blocks);
- defines effective-dimension proxies tied to RLCT, based on Fisher spectra and learning curves;
- specifies OCTA-flavored protocols for integrating these measurements into training loops ("OCTA-Fisher Pipeline");

- proposes visualizations and dashboards for tracking singular geometry during training and scaling experiments;
- introduces curvature regimes, OCTA Principles, and curvature-based alerts as safety-relevant signals;
- includes a simple analytical toy model where Fisher and curvature patterns can be computed explicitly;
- and prepares the ground for Day 5, which will link these probes to empirical scaling laws and capability transitions.

The operational goal: treat *Fisher geometry and RLCT proxies* as first-class diagnostics in OCTA-scale AGI systems, with clear interfaces to engineering decisions.

Contents

List of Figures

1 From SLT Theory (Day 3) to Practical Probes

Day 3 established:

- the KL divergence $K(\theta)$ as the central statistical functional;
- the RLCT λ as the effective dimension governing generalization and Bayesian free energy;
- the role of attention-induced singular sets in shaping the local algebraic structure of $K(\theta)$;
- qualitative links between Fisher degeneracy and attention strata.

Day 4 addresses the practical question:

How do we measure and monitor these quantities in a real, large Transformer?

We do not attempt to compute exact RLCT (which would require full algebraic desingularization). Instead, we define:

- consistent definitions for population Fisher, empirical Fisher, Gauss–Newton, and Hessian;
- scalable estimators for eigenvalue spectra and trace/log-determinant-like quantities;
- practical proxies for RLCT, effective dimension, and singularity strength;
- instrumentation protocols that can run alongside standard training.

Throughout, we emphasize OCTA integration: these probes are intended to become standard metrics in the system, not experimental afterthoughts.

2 Population Fisher, Empirical Fisher, and Hessian

Let $p_\theta(y \mid x)$ be a Transformer model and $q(x, y)$ the true data distribution.

2.1 Population Fisher

Definition 2.1 (Population Fisher)**.** The *Fisher information matrix* at θ is

$$F(\theta) := \mathbb{E}_{(x,y)\sim q}\left[\nabla_\theta \log p_\theta(y \mid x)\ \nabla_\theta \log p_\theta(y \mid x)^\top\right]. \tag{1}$$

In regular models, $F(\theta^\star)$ is nonsingular at the KL minimizer $\theta^\star$; in singular models it is generically rank-deficient at every $\theta \in \Theta^\star$.

2.2 Empirical Fisher

Given a dataset $D_n = \{(x_i, y_i)\}_{i=1}^n$, the *empirical Fisher* is

$$\hat{F}_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} g_i(\theta) g_i(\theta)^\top, \quad g_i(\theta) := \nabla_\theta \log p_\theta(y_i \mid x_i). \tag{2}$$

In practice, we typically compute minibatch approximations

$$\hat{F}_B(\theta) := \frac{1}{|B|}\sum_{i\in B} g_i(\theta) g_i(\theta)^\top, \tag{3}$$

for a minibatch B.

2.3 Hessian and Gauss–Newton

For a negative log-likelihood loss

$$\ell(\theta) = -\log p_\theta(y \mid x),$$

the Hessian is

$$H(\theta) = \nabla_\theta^2 \ell(\theta).$$

For generalized linear models and some neural networks, one may approximate $H(\theta)$ with a *Gauss–Newton* matrix

$$G(\theta) = J(\theta)^\top W J(\theta),$$

where J is the Jacobian of outputs with respect to parameters and W is a positive semi-definite weighting matrix (e.g., based on output variance).

In many practical regimes, F, G, and H share leading eigen-directions and eigenvalues; however, they can differ materially in highly non-linear or singular regimes. Day 4 focuses primarily on F and its empirical approximations, while noting where Gauss–Newton proxies are more accessible.

3 Scalable Estimation in Large Transformers

Computing $F(\theta)$ exactly is infeasible for large models: θ has dimension d in the millions to billions, and F is a $d \times d$ matrix. We therefore rely on approximations.

We outline three levels of sophistication:

- **Level 1:** scalar summaries (traces and norms);
- **Level 2:** low-rank eigen-slices of the spectrum;
- **Level 3:** structured block approximations (layerwise, headwise).

3.1 Level 1: Hutchinson trace and norms

Let A be a large symmetric matrix accessible only via matrix-vector products Av. The *Hutchinson estimator* for $\mathrm{tr}(A)$ is

$$\mathrm{tr}(A) = \mathbb{E}_z\left[z^\top A z\right], \tag{4}$$

where z has i.i.d. Rademacher (± 1) or standard Gaussian entries. In practice, we use

$$\widehat{\mathrm{tr}}(A) = \frac{1}{N_z} \sum_{k=1}^{N_z} z_k^\top A z_k. \tag{5}$$

To apply this to the empirical Fisher:

- implement a routine that, given v, computes $Fv \approx \hat{F}_B(\theta)v$ using a minibatch B and automatic differentiation:
$$Fv \approx \frac{1}{|B|} \sum_{i \in B} (g_i^\top v) g_i; \tag{6}$$
- apply Hutchinson's method to estimate $\mathrm{tr}(F)$, which measures total curvature energy;
- analogously estimate $\mathrm{tr}(F^2)$ or higher moments if needed via repeated products.

These scalar summaries are cheap and can be logged throughout training as OCTA diagnostics.

3.2 Level 2: Low-rank eigen-slices

To capture shape, not just total energy, we approximate leading and/or trailing eigenvalues and eigenvectors using iterative methods (Lanczos, power iteration).

Schematic procedure:

(a) Implement a function `FisherMv(theta, v)` that returns Fv via minibatch gradient accumulation.

(b) Use a Lanczos or power-iteration routine with `FisherMv` as the matrix-vector oracle.

(c) Extract:
   - top-k eigenpairs $(\lambda_j^{\mathrm{top}}, u_j^{\mathrm{top}})$;
   - optionally, bottom-k eigenpairs via shift-invert or deflated iteration.

(d) Log the spectrum and track its evolution across training.

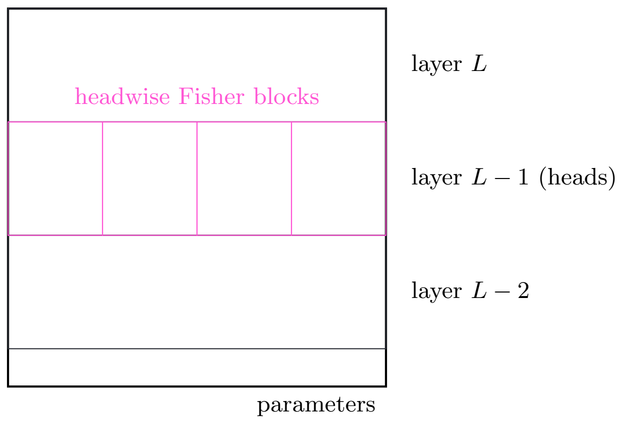

Figure 1: Structured block view of Fisher: block-diagonal by layer, with headwise blocks inside attention layers. This aligns with Day 2 attention strata and makes curvature diagnostics interpretable at the level of individual heads.

In practice:

- k can be as small as 10–50 to reveal structure;
- these computations can be performed periodically (e.g., every K training steps) to limit overhead.

3.3 Level 3: Structured block approximations (layerwise, headwise)

Full-matrix eigen computations remain expensive even with iterative methods, especially if we want layerwise detail.

We therefore consider structured factorizations:

- *Layerwise Fisher blocks:* block-diagonal approximations where each block corresponds to a transformer layer;
- *Headwise Fisher blocks:* finer decomposition where each attention head is treated as a separate block;
- *Kronecker factorizations (K-FAC style):* approximate per-layer Fisher as Kronecker products of smaller matrices (e.g., input covariance $\otimes$ gradient covariance);
- *Hybrid*: layerwise for MLPs, headwise for attention, all feeding into an OCTA dashboard.

For OCTA, this naturally aligns with the Day 2 decomposition:

- per-head analysis of attention strata and tie-manifold proximity;
- per-layer analysis of block cells and MLP activations.

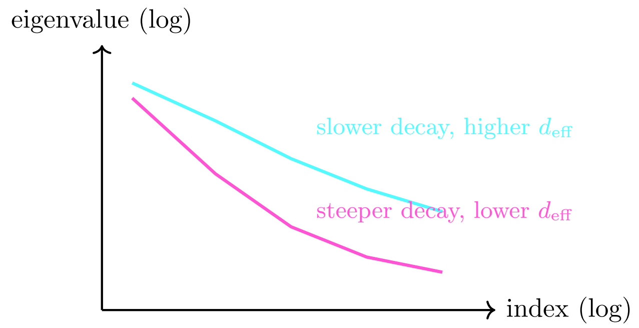

Figure 2: Log–log Fisher eigenvalue spectra. The slope and curvature of the spectrum provide a diagnostic for effective dimension and anisotropy.

3.4 Headwise Fisher and attention strata

For an attention head h with parameters $\theta^{(h)}$, define a headwise Fisher block

$$F^{(h)}(\theta) = \mathbb{E}\left[\nabla_{\theta^{(h)}} \log p_\theta(y \mid x)\, \nabla_{\theta^{(h)}} \log p_\theta(y \mid x)^\top\right]. \tag{7}$$

We can interpret:

- large eigenvalues of $F^{(h)}$ as directions where that head is *statistically influential*;
- near-zero eigenvalues as directions where that head is redundant, tied to other heads, or stuck in attention plateaus.

Overlaying this with Day 2:

- heads that frequently operate near tie manifolds and share patterns with other heads may have highly singular $F^{(h)}$;
- OCTA can tag such heads as "degenerate" vs. "structurally active" based on combined curvature and strata-visitation signals.

3.5 Spectrum visualization: log–log diagnostics

Beyond a few eigenvalues, it is useful to plot spectra on log–log axes:

- slowly decaying spectra indicate many active directions and higher effective dimension;
- sharply decaying spectra indicate strong anisotropy and lower effective dimension.

4 Effective Dimension and RLCT Proxies

Exact RLCT λ is difficult to compute. However, Fisher-based quantities and learning-curve fits can serve as practical proxies.

4.1 Fisher-based effective dimension

Given eigenvalues $\{\lambda_j\}_{j=1}^d$ of the (population or empirical) Fisher matrix $F(\theta)$, consider:

(i) Ridge-regularized effective dimension. For a regularization parameter $\alpha > 0$, define

$$d_{\text{eff}}(\alpha) = \sum_{j=1}^{d} \frac{\lambda_j}{\lambda_j + \alpha}. \tag{8}$$

This is analogous to effective degrees of freedom in kernel methods and linear models.
Interpretation:

- directions with $\lambda_j \gg \alpha$ contribute ≈ 1;
- directions with $\lambda_j \ll \alpha$ contribute ≈ 0.

(ii) Trace-based surrogate. If only the trace is available, one may use

$$\tilde{d}_{\text{eff}}(\alpha) = \frac{\operatorname{tr}(F)}{\alpha + \bar{\lambda}}, \tag{9}$$

where $\bar{\lambda}$ is a suitable average eigenvalue (e.g. $\operatorname{tr}(F)/d$). This is crude but correlates with concentration of curvature.

(iii) Entropic spectrum complexity. Define a normalized spectrum

$$\tilde{\lambda}_j = \frac{\lambda_j}{\sum_k \lambda_k},$$

and an entropy

$$H_{\text{spec}} = -\sum_j \tilde{\lambda}_j \log \tilde{\lambda}_j. \tag{10}$$

Low entropy (few dominant eigenvalues) indicates strong anisotropy and potentially lower effective dimension.

4.2 Curvature regimes and RLCT interpretation

From Day 3, near a KL minimizer in a singular model, the leading generalization term is λ/n. Empirically, we can think in terms of *curvature regimes*.

Definition 4.1 (Curvature regimes)**.** Let $\lambda_{\max}$ denote the largest eigenvalue of an empirical Fisher matrix at some stage of training, and let

$$\rho = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

be the fraction of total curvature captured by the top-k eigenvalues for some fixed k. We define:

- *Flat regime:* $\lambda_{\max}$ small, ρ moderate, H_{spec} high.
- *Intermediate regime:* $\lambda_{\max}$ moderate, ρ neither extremely small nor extremely large, spectrum decays with a visible elbow.
- *Sharp regime:* $\lambda_{\max}$ large and/or ρ very high for small k, H_{spec} low.

OCTA Principle 4.2 (OCTA Curvature Principle I – Balanced Regime). In OCTA training pipelines, *intermediate curvature regimes* are preferred:

- sufficiently large $d_{\text{eff}}(\alpha)$ and λ_{eff} to express the task;
- but spectra with controlled $\lambda_{\max}$ and non-degenerate tail behavior, avoiding extreme concentration.

Moving systematically toward or away from this regime can be used as a control signal in architecture and hyperparameter search.

OCTA Principle 4.3 (OCTA Curvature Principle II – Safety Surface). Curvature metrics (e.g. $\lambda_{\max}$, headwise traces, H_{spec}, λ_{eff}) define a *safety surface*: regions of parameter space where

- $\lambda_{\max}$ is excessive,
- curvature concentrates heavily in a small number of heads or layers,
- or λ_{eff} changes too abruptly with n

should be treated as candidates for:

- learning-rate reduction,
- gradient clipping / regularization increase,
- or architecture adjustment (e.g. head tying, capacity redistribution).

4.3 Learning-curve-based RLCT proxies

Day 3 showed that in singular models

$$\mathbb{E}[G_n] \sim \frac{\lambda}{n}.$$

In practice, we can estimate λ by fitting learning curves.

For dataset sizes n_k, measure test negative log-likelihood or generalization gaps G_{n_k} and fit

$$G_{n_k} \approx \frac{\lambda_{\text{eff}}}{n_k} + \frac{b}{n_k^{\alpha}}. \tag{11}$$

The fitted λ_{eff} serves as an RLCT proxy; importantly, we can:

- track how λ_{eff} changes with architecture modifications;
- compare λ_{eff} to Fisher-based $d_{\text{eff}}(\alpha)$;
- use discrepancies as signals of non-asymptotic regimes or model mismatch.

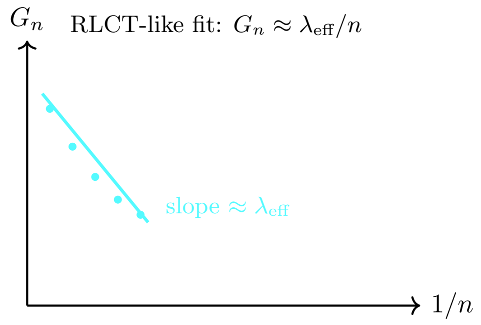

Figure 3: Learning-curve-based RLCT proxy. Plotting generalization gap G_n vs $1/n$ and fitting a line yields an effective RLCT estimate λ_{eff} in the asymptotic regime.

4.4 OCTA RLCT scoreboard

We propose an internal *OCTA RLCT Scoreboard* with metrics:

- $d_{\text{eff}}(\alpha)$ at several α values (e.g. $\alpha \in \{10^{-4}, 10^{-3}, 10^{-2}\}$);
- top-k Fisher eigenvalues and their ratios;
- spectrum entropy H_{spec};
- fitted λ_{eff} from learning curves;
- layerwise/headwise curvature summaries (e.g. per-layer traces and eigen-slices);
- flags for curvature regimes (flat, intermediate, sharp).

5 OCTA Protocols for Fisher and RLCT Instrumentation

We now define concrete OCTA protocols that extend Day 3's D3.1–D3.5 with implementation details.

5.1 Protocol D4.1: Training-loop Fisher snapshots

Experimental Protocol 5.1 (D4.1: Periodic Fisher snapshots during training).

(a) Choose a set of training steps $\{t_1, \dots, t_m\}$ at which to take Fisher snapshots (e.g. logarithmically spaced, with extra density around known capability transitions).

(b) At each t_k:

  (i) freeze current parameters θ_{t_k};
  (ii) draw a diagnostic minibatch B_k (distinct from training batch if desired);
  (iii) compute an empirical Fisher oracle $v \mapsto \hat{F}_{B_k}(\theta_{t_k})v$;
  (iv) estimate:
    - trace $\operatorname{tr}(\hat{F}_{B_k})$ via Hutchinson;
    - leading eigenvalues via k-step power iteration or Lanczos;
    - optionally, layerwise/headwise block traces.
  (v) log results to the OCTA RLCT scoreboard.

5.2 Protocol D4.2: Multi-n training for RLCT curves

Experimental Protocol 5.2 (D4.2: Multi-n training for RLCT proxy).

(a) Select dataset size multipliers $c_1 < c_2 < \cdots < c_m$ and a base size n_0.

(b) For each k, define $n_k = c_k n_0$ and:

- either subsample a dataset of size n_k,
- or use a controllable data loader that emulates size n_k.

(c) For each n_k, train a model (or fine-tune from a shared initialization) to near convergence.

(d) Measure test negative log-likelihood and compute generalization gaps G_{n_k}.

(e) Fit G_{n_k} vs $1/n_k$ to estimate λ_{eff}.

(f) Compare λ_{eff} across architectures and training setups.

5.3 Protocol D4.3: Fisher–strata joint logging

Experimental Protocol 5.3 (D4.3: Joint Fisher and attention-strata logging).

(a) At Fisher snapshot steps t_k (Protocol D4.1), additionally:

- for each input in a diagnostic set, record:
  - attention patterns σ per head;
  - block cell IDs (σ, a) (attention pattern + activation pattern) as defined in Day 2;
  - distances to tie manifolds (Day 2 Protocols).

(b) Aggregate:

- distribution of visited strata;
- average curvature (Fisher trace) per layer/head;
- correlation between strata visitation and local Fisher eigenvalues.

(c) Visualize:

- a heatmap of curvature vs. strata occupancy,
- evolution of this heatmap across training steps.

5.4 Protocol D4.4: OCTA-Fisher dashboard

Experimental Protocol 5.4 (D4.4: OCTA-Fisher dashboard).

(a) Implement a dashboard (internal UI or logging schema) that, for each run, displays:

- training/validation loss and accuracy;
- trace of Fisher vs training step;
- top-k eigenvalues and spectrum entropy;
- per-layer/per-head Fisher traces and heatmaps;

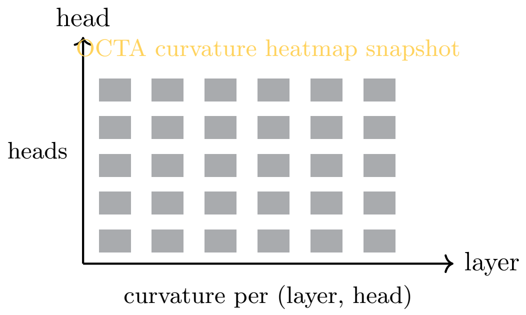

Figure 4: Conceptual curvature heatmap: each cell corresponds to (layer, head) and is colored by a curvature measure (e.g. Fisher trace per head). This can be jointly analyzed with attention-strata occupancy.

- RLCT proxy λ_{eff} vs parameter count and dataset size;
- attention-strata distributions and their evolution.

(b) Use this dashboard for:

- architecture comparison,
- training regime debugging (e.g. too-sharp curvature spikes),
- safety/robustness assessment (e.g. overly concentrated curvature in certain heads),
- anomaly detection during large-scale OCTA deployments.

5.5 Protocol D4.5: Curvature-based safety and stability alerts

Experimental Protocol 5.5 (D4.5: Curvature-based alert system).

(a) Define thresholds:

- $\lambda_{\max}^{\text{crit}}$ for top eigenvalue;
- $\Delta\lambda_{\max}^{\text{crit}}$ for sudden increases between snapshots;
- $r_{\text{head}}^{\text{crit}}$ for headwise curvature concentration (e.g. fraction of total trace in one head).

(b) At each Fisher snapshot:

- compute $\lambda_{\max}$ and headwise curvature fractions;
- if $\lambda_{\max} > \lambda_{\max}^{\text{crit}}$ or $\Delta\lambda_{\max} > \Delta\lambda_{\max}^{\text{crit}}$, raise an alert;
- if any head exceeds $r_{\text{head}}^{\text{crit}}$ of total curvature, mark that head as a potential instability source.

(c) Integrate with training control:

- optionally apply learning rate reductions, gradient clipping, or regularization adjustments when alerts fire;
- log alerts and subsequent behavior for postmortem analysis.

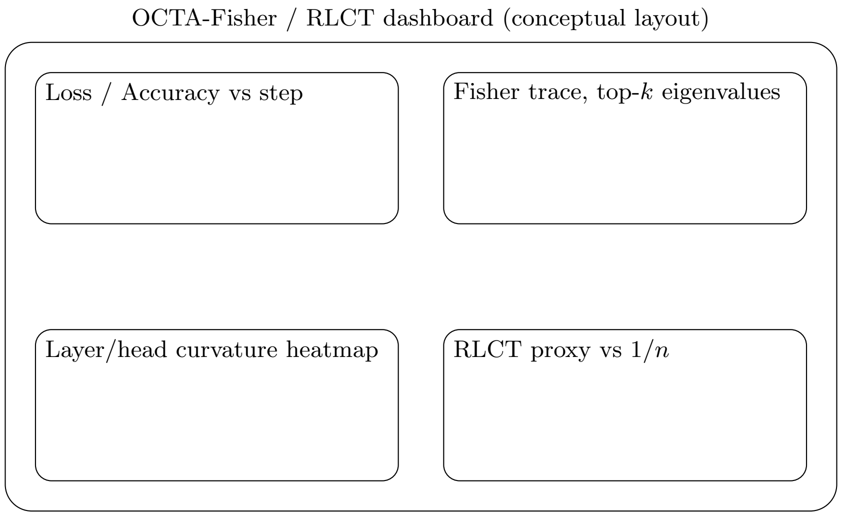
OCTA-Fisher / RLCT dashboard (conceptual layout)
Loss / Accuracy vs step
Fisher trace, top-k eigenvalues
Layer/head curvature heatmap
RLCT proxy vs $1/n$

Figure 5: Conceptual layout of an OCTA-Fisher / RLCT dashboard. Top: standard metrics plus Fisher trace and eigenvalues; bottom: layer/head curvature heatmap and RLCT proxy fits vs dataset size.

5.6 Protocol D4.6: RLCT-aware architecture comparison

Experimental Protocol 5.6 (D4.6: RLCT-aware architecture evaluation).

(a) For a set of candidate architectures (varying depth, width, head count, tying), run D4.2 to estimate λ_{eff} and D4.1 to measure Fisher spectra.

(b) For each architecture, compute:

- λ_{eff} (learning-curve RLCT proxy);
- $d_{\text{eff}}(\alpha)$ for several α;
- curvature regime classification (flat / intermediate / sharp).

(c) Select architectures where:

- λ_{eff} is large enough to model the task but not excessively large relative to d;
- Fisher spectra exhibit intermediate regimes rather than extremely sharp or overly flat behavior;
- headwise curvature is not excessively concentrated.

6 Implementation Skeletons (Pseudo-code)

We include simplified pseudo-code sketches for key primitives; actual implementations will depend on the deep learning framework.

6.1 Fisher matrix-vector product

```
function fisher_mv(theta, v, dataloader, num_batches):
    # theta: current parameters (flattened)
    # v: vector in parameter space (flattened)
    # dataloader: yields (x, y) minibatches
    # num_batches: number of batches to average over
    acc = zeros_like(theta)
    count = 0
    for (x, y) in dataloader:
        if count == num_batches:
            break
        # 1. Forward pass to compute log p_theta(y | x)
        loss = -log_prob(theta, x, y)  # scalar loss
        # 2. Compute gradient g = grad_theta log p
        g = grad(loss, theta) * (-1.0)  # since loss = -log p
        # 3. Compute scalar (g^T v)
        s = dot(g, v)
        # 4. Accumulate s * g
        acc += s * g
        count += 1
    acc /= max(count, 1)
    return acc   # approximates F v
```

6.2 Hutchinson trace estimator

```
function fisher_trace(theta, dataloader, num_batches, num_vecs):
    trace_est = 0.0
    for k in range(num_vecs):
        z = random_rademacher_like(theta)  # +/- 1 entries
        Fz = fisher_mv(theta, z, dataloader, num_batches)
        trace_est += dot(z, Fz)
    trace_est /= max(num_vecs, 1)
    return trace_est
```

6.3 Power iteration for top eigenvalue

```
function top_eigenpair(theta, dataloader, num_batches, num_iters):
    v = random_normal_like(theta)
    v = v / norm(v)
    for t in range(num_iters):
        w = fisher_mv(theta, v, dataloader, num_batches)
        norm_w = norm(w) + 1e-12
        v = w / norm_w
    lambda_top = dot(v, fisher_mv(theta, v, dataloader, num_batches))
    return lambda_top, v
```

6.4 Layerwise block Fisher estimation

```
function fisher_layer_traces(theta, dataloader, num_batches, num_vecs, layer_slices):
    # layer_slices: mapping layer_id -> indices in flattened theta
    traces = dict()
    for layer_id, idx in layer_slices.items():
        traces[layer_id] = 0.0

    for k in range(num_vecs):
        z_full = random_rademacher_like(theta)
        Fz_full = fisher_mv(theta, z_full, dataloader, num_batches)
        for layer_id, idx in layer_slices.items():
            z = z_full[idx]        # restrict to layer block
            Fz = Fz_full[idx]
            traces[layer_id] += dot(z, Fz)

    for layer_id in traces:
        traces[layer_id] /= max(num_vecs, 1)
    return traces
```

6.5 Headwise block Fisher estimation

```
function fisher_head_traces(theta, dataloader, num_batches, num_vecs, head_slices):
    # head_slices: mapping (layer_id, head_id) -> indices in flattened theta
    traces = dict()
    for key, idx in head_slices.items():
        traces[key] = 0.0

    for k in range(num_vecs):
        z_full = random_rademacher_like(theta)
        Fz_full = fisher_mv(theta, z_full, dataloader, num_batches)
        for key, idx in head_slices.items():
            z = z_full[idx]
            Fz = Fz_full[idx]
            traces[key] += dot(z, Fz)

    for key in traces:
        traces[key] /= max(num_vecs, 1)
    return traces
```

6.6 Kronecker-style per-layer approximation (skeleton)

```
function kfac_layer_fisher_stats(model, dataloader, num_batches):
    # Pseudo-code for single layer (e.g., linear or attention projection)
    A_cov = None
    G_cov = None
    count = 0
    for (x, y) in dataloader:
        if count == num_batches:
            break
        # Forward to get activations a and pre-activation gradients g
        a, g = forward_and_backprop_layer(model, x, y)
        # Accumulate covariances
        if A_cov is None:
            A_cov = outer(a, a)
            G_cov = outer(g, g)
        else:
            A_cov += outer(a, a)
            G_cov += outer(g, g)
        count += 1
    if count == 0:
        return None, None
    A_cov /= count
    G_cov /= count
    # Approximate layer Fisher ~ kron(G_cov, A_cov)
    return A_cov, G_cov
```

These sketches specify the core primitives needed to implement the OCTA-Fisher pipeline of Figure 5, plus layerwise and Kronecker-style diagnostics aligned with attention and MLP blocks.

7 Natural Gradient and Preconditioning

The same Fisher information that provides RLCT proxies also underlies *natural gradient* methods.

Definition 7.1 (Natural gradient step)**.** Given an update direction $g = \nabla_\theta L(\theta)$ and Fisher matrix $F(\theta)$, the natural gradient update is

$$\Delta\theta_{\text{nat}} = -\eta\, F(\theta)^{-1} g,$$

for learning rate $\eta > 0$ (assuming F is regularized to be invertible).

In singular or near-singular regimes:

- F is not invertible; we instead use a pseudo-inverse or regularized inverse $(F + \alpha I)^{-1}$;
- curvature diagnostics from Day 4 directly determine which directions are amplified or suppressed by F^{-1}.

OCTA Principle 7.2 (OCTA Curvature Principle III – Geometry-aware Preconditioning). OCTA training loops should treat Fisher-based curvature not only as a metric but as a *preconditioning resource*:

- use blockwise Fisher approximations (e.g. Kronecker factors) to construct low-rank or layerwise preconditioners;
- tune regularization α based on curvature regimes (larger in sharp regimes);
- monitor the interaction of preconditioning with RLCT proxies to avoid overly aggressive steps that push the model into sharp, unstable regions.

This connects Day 4 diagnostics to concrete optimization choices: natural gradient variants, K-FAC, Shampoo, or other second-order methods can be aligned with the same Fisher infrastructure that feeds the OCTA dashboard.

8 Toy Example: One-Dimensional Curvature and RLCT Proxy

To anchor intuition, we consider a simple one-parameter model where Fisher and learning curves can be analyzed explicitly.

8.1 Setup

Let $x \in \mathbb{R}$ be input, $y \in \{0, 1\}$ a binary label, and consider a scalar parameter $\theta \in \mathbb{R}$ with logistic regression

$$p_\theta(y = 1 \mid x) = \sigma(\theta x), \quad p_\theta(y = 0 \mid x) = 1 - \sigma(\theta x),$$

where $\sigma(z) = 1/(1+e^{-z})$. Assume x has distribution q_X and labels follow $y \mid x \sim \text{Bernoulli}(p_{\theta^\star}(1 \mid x))$ for some true parameter $\theta^\star$.

This model is regular, not singular, but it gives a clean baseline.

8.2 Fisher information

The log-likelihood for a single example is

$$\log p_\theta(y \mid x) = y \log \sigma(\theta x) + (1 - y) \log(1 - \sigma(\theta x)).$$

We compute

$$\frac{\partial}{\partial \theta} \log p_\theta(y \mid x) = (y - \sigma(\theta x))x.$$

Thus,

$$F(\theta) = \mathbb{E}_{(x,y)\sim q} \left[(y - \sigma(\theta x))^2 x^2\right].$$

Taking expectation over $y \mid x$,

$$\mathbb{E}\big[(y - \sigma(\theta x))^2 \mid x\big] = \sigma(\theta x)\big(1 - \sigma(\theta x)\big),$$

so

$$F(\theta) = \mathbb{E}_{x\sim q_X} \left[\sigma(\theta x)\big(1 - \sigma(\theta x)\big)x^2\right]. \tag{12}$$

At $\theta = \theta^\star$, this is a positive scalar as long as x is not almost surely zero.

8.3 Effective dimension and RLCT

In this one-dimensional regular model:

- the Fisher matrix is just the scalar $F(\theta^\star)$;
- the effective dimension $d_{\text{eff}}(\alpha)$ is

$$d_{\text{eff}}(\alpha) = \frac{F(\theta^\star)}{F(\theta^\star) + \alpha},$$

  which is close to 1 for $\alpha \ll F$;
- the RLCT is $\lambda = 1/2$ for the parameter θ in a standard regular setting (in Watanabe's conventions, effective dimension is $1/2$ per scalar parameter for certain formulations), or $d/2$ with $d = 1$.

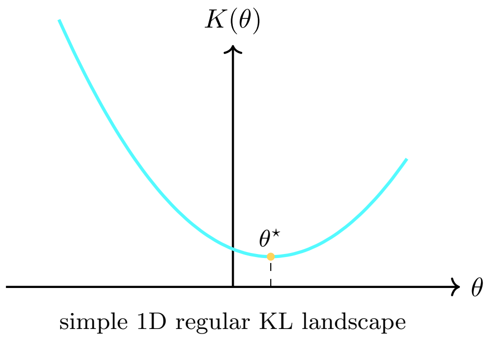

Figure 6: Toy 1D example: in a regular model, the KL divergence $K(\theta)$ is locally quadratic around $\theta^\star$, and Fisher is a positive scalar. In Transformers, the local geometry is higher-dimensional and singular; Day 4 measures this geometry via Fisher spectra.

Remark 8.1. In this simple model, Fisher, Hessian, $d_{\text{eff}}(\alpha)$, and λ all agree with the intuition that there is a single active direction. For Transformers, Day 4 generalizes this structure: Fisher is high-dimensional, rank-deficient, and shaped by attention strata; RLCT captures the effective number of degrees of freedom after singularities are accounted for.

This toy example shows how Fisher and RLCT are aligned in the simplest case; Transformers inherit the same fundamental relationship but in a dramatically more complex, singular landscape that Day 4 aims to instrument.

9 OCTA Integration Checklist

To operationalize Day 4 in the OCTA ecosystem, we can use the following checklist per project or architecture:

- **Instrumentation:**
  - ☐ Implement `fisher_mv`, `fisher_trace`, and `top_eigenpair` for the main training framework.
  - ☐ Implement layerwise/headwise index slices for blockwise traces.
  - ☐ Optional: implement K-FAC-style per-layer stats for MLPs and attention projections.
- **Logging:**
  - ☐ Schedule Fisher snapshots (D4.1) along training steps, including around suspected capability transitions.
  - ☐ Log Fisher traces, top-k eigenvalues, and spectrum entropy at each snapshot.
  - ☐ Log per-layer and per-head curvature traces and associate them with attention-strata statistics (D4.3).
- **RLCT Proxies:**
  - ☐ Run multi-n experiments (D4.2) for at least one representative architecture to calibrate λ_{eff}.
  - ☐ Compare λ_{eff} with Fisher-based $d_{\text{eff}}(\alpha)$ across architectures.

- **Dashboard:**
  - ☐ Set up an OCTA-Fisher dashboard (D4.4) with panels for:
    - ∗ loss/accuracy vs. step,
    - ∗ Fisher traces/eigenvalues,
    - ∗ curvature heatmaps (layer/head),
    - ∗ RLCT vs. $1/n$,
    - ∗ attention strata distributions.
- **Safety Hooks:**
  - ☐ Implement curvature-based alerts (D4.5) feeding into training controls (learning rate, clipping, regularization).
  - ☐ Record alerts and post-alert behavior for safety analysis and future policy tuning.
- **Architecture Tuning:**
  - ☐ For each new architecture family, run RLCT-aware comparison (D4.6) to select design points with acceptable curvature regimes.
  - ☐ Use curvature regime classification (flat/intermediate/sharp) as a filtering criterion before large-scale deployment.
- **Preconditioning:**
  - ☐ Optionally integrate blockwise Fisher approximations into natural-gradient or K-FAC-style optimizers.
  - ☐ Tune preconditioner regularization using curvature regimes from the scoreboard.

This checklist turns Day 4 from theoretical tooling into a concrete operational standard for OCTA runs.

10 Connection to Scaling Laws (Preview of Day 5)

Empirical scaling laws for language models suggest power-law relationships between:

- loss and compute,
- loss and dataset size,
- loss and model size.

Day 3 argued that RLCT λ controls the $1/n$ term in the asymptotic expansion of generalization error. Day 4 provides tools to:

- measure effective dimension via Fisher geometry;
- estimate RLCT-like coefficients from learning curves;
- track curvature regimes, safety alerts, and headwise singularity as model and dataset scales change.

Day 5 will:

- connect these measurements to empirically observed scaling exponents;
- analyze deviations from simple $1/n$ behavior (e.g. multiple regimes, phase transitions);
- explore how attention stratification and singular geometry correlate with scaling behavior and emergent capabilities;
- outline how OCTA can deliberately *steer* scaling trajectories using curvature and RLCT diagnostics as control signals.

For OCTA, the strategic objective is to treat scaling-law exponents, Fisher spectra, and RLCT proxies as a *unified diagnostic surface* on which we can steer architectures and training policies.

11 Conclusion

Day 4 has:

- clarified the roles of population Fisher, empirical Fisher, Gauss–Newton, and Hessian for Transformer models;
- outlined scalable methods to estimate Fisher traces, eigen-slices, and structured blocks using minibatch gradients and matrix-vector oracles;
- introduced headwise and layerwise Fisher blocks aligned with Day 2's attention strata and block cells;
- defined Fisher-based effective-dimension metrics and learning-curve-based RLCT proxies, along with explicit curvature regimes (flat, intermediate, sharp);
- established OCTA Principles connecting curvature patterns to training quality and safety surfaces;
- proposed an OCTA RLCT scoreboard aggregating these metrics for each run;
- specified practical OCTA protocols (D4.1–D4.6) for integrating Fisher and RLCT instrumentation into training loops, dashboards, safety alerts, and architecture selection;
- connected Fisher geometry to natural gradient and preconditioning choices;
- and provided a toy example illustrating how Fisher and RLCT behave in the simplest regular case, as a baseline for the Transformer setting.

Together with Days 1–3:

- Day 1 provides the macro vs. micro singularity frame;
- Day 2 provides the attention and block-cell geometry;
- Day 3 provides the statistical singularity theory and RLCT;
- Day 4 provides the measurement tools, safety hooks, optimization links, and integration pipeline.

The next step (Day 5) will leverage these tools to analyze and shape scaling laws, linking singular geometry, effective dimension, and emergent capabilities in the OCTA ecosystem.

References

[1] S. Amari and H. Nagaoka. *Methods of Information Geometry.* AMS and Oxford University Press, 2000.

[2] J. Kaplan, S. McCandlish, T. Henighan, et al. Scaling laws for neural language models. arXiv:2001.08361, 2020.

[3] J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, 2015.

[4] R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. arXiv:1301.3584, 2014.

[5] S. Watanabe. *Algebraic Geometry and Statistical Learning Theory.* Cambridge University Press, 2009.

[6] S. Watanabe. *Mathematical Theory of Bayesian Statistics.* CRC Press, 2018.