Outcome Prediction Quiz: Stability

CS 312 · 40 minutes · 100 points

How to answer. This quiz has six questions. Put your name, SUNet ID, choices, and explanations on the separate answer sheet at the end. Only that sheet is graded.

Use your assignment experiments to predict the outcomes. Detailed experimental evidence and sound reasoning can recover up to 75% of missed points. Adjacent loss-regret intervals earn half credit. Each question points to its code diff in Appendix A.

Reference recipe. The released A4 recipe uses width 512, vocabulary 4096, context 1024, batch 64, 600,000 fresh training sequences (614.4M token slots; 9,375 updates), AdamW, linear LR decay with 1% warmup, masked weight decay .1, clipping at 1, and model/data seed 42. Lower validation loss is better.

The transformer is the default 8-block model with RMSNorm, QK normalization, RoPE, and SwiGLU MLPs. The **Modern tanh baseline** is the released four-block `ModernRNN`: each block contains a tanh recurrence and a SwiGLU MLP, both with depth residuals and pre-LayerNorm. The GRU keeps this depth stack and changes only the recurrent update.

Unless stated otherwise, train each variant from scratch for all 9,375 updates. Select the run with the lowest *final* validation loss on that variant's released/reference LR grid; do not select an earlier training checkpoint. All runs use **linear decay** and the other defaults above.

Diagnostics use common held-out inputs and FP32. Depth activations are complete block outputs, before the readout norm; embeddings are excluded. Loss is mean cross-entropy per next-token target.

Code reference. The diffs apply independently to the released A4 starter at `4350166`. Questions 1–3 change the measurements; Questions 4–6 add model variants. Training and execution backends remain those of the release.

Q1. Norms across depth

Difficulty: Easy

12 points

From A4 Problem 1. Match the three trained transformer variants to the activation plots in (a) and gradient plots in (b). Use A–C once in each part; the two parts use different letter assignments.

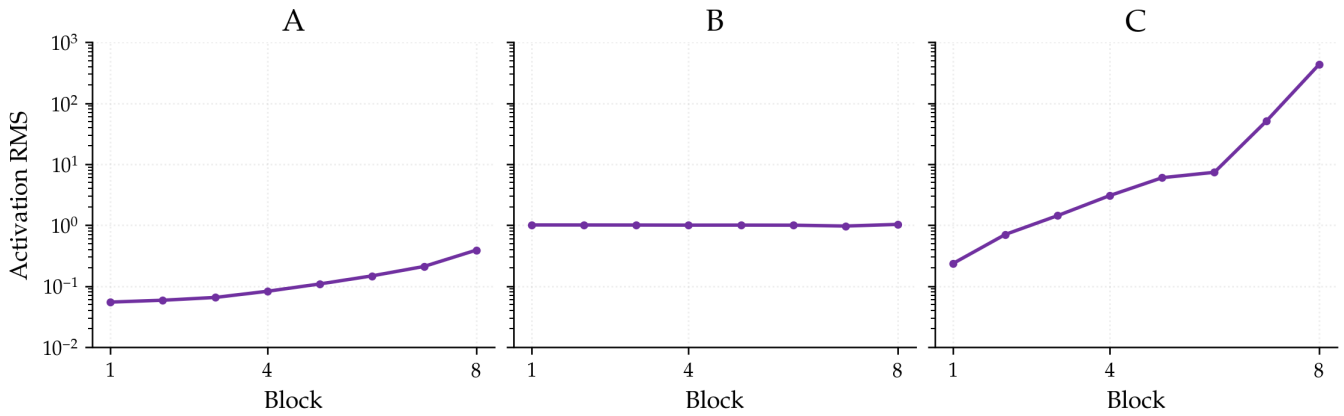
For each attention or MLP sublayer, x is the incoming depth stream, F its operation, and N its RMSNorm. The three updates are:

$$\begin{aligned} &\text{No pre-norm:} && x' = x + F(x), \\ &\text{Post-norm:} && x' = N(x + F(x)), \\ &\text{No residual:} && x' = F(N(x)). \end{aligned}$$

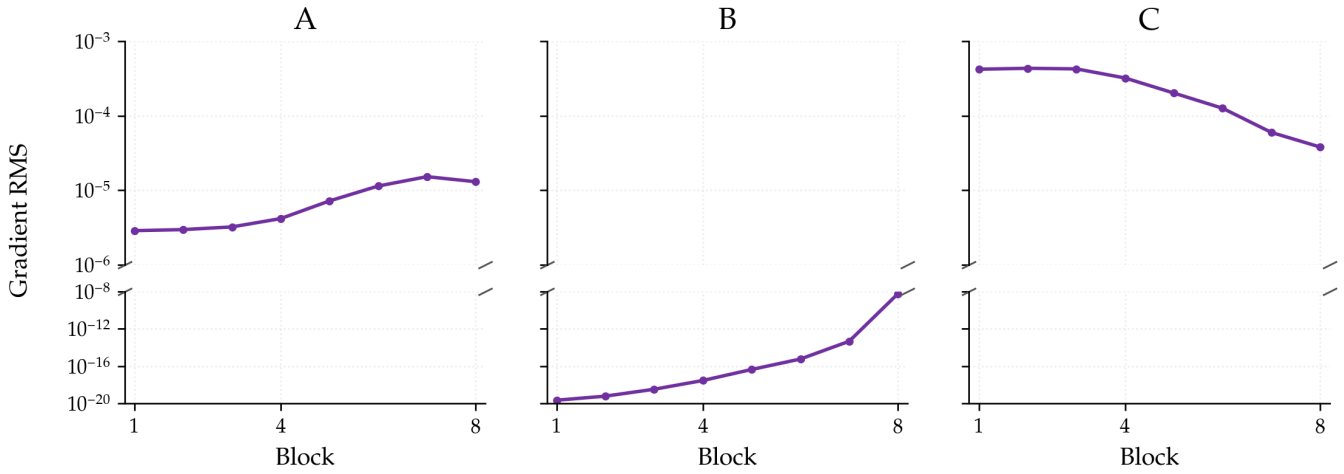
Each variant changes both attention and MLP sublayers. At each complete block output x_ℓ, measure $\mathrm{RMS}(x_\ell)$ and $\mathrm{RMS}(\partial L/\partial x_\ell)$ over batch, time, and features. L is the mean loss over all next-token targets.

Diff against the A4 release: Appendix A.1, page 8.

(a) Activation RMS (6 points; 2 per variant).



(b) Activation-gradient RMS (6 points; 2 per variant). The shared logarithmic y-axis is broken between 10^{-8} and 10^{-6}.



Q2. RNNs and transformers across depth

Difficulty: Medium

16 points

From A4 Problems 1–2. Compare the four-block Modern tanh baseline and default eight-block transformer. Train from scratch with the released recipe and the learning rates specified below.

Diff against the A4 release: Appendix A.2, page 8.

(a) Stream growth (4 points). Use the released default LRs: .003 for the transformer and .001 for the RNN, without tuning. Let x_ℓ be the stream after complete block ℓ, including its MLP and depth skips. On the same four held-out sequences, measure

$$R = \frac{\mathrm{RMS}(x_4)}{\mathrm{RMS}(x_1)} \qquad \text{for both models.}$$

RMS averages over batch, time, and features. Which model has the larger R from layer 1 to layer 4? Choose **Within 50%** if $\max(R_\mathrm{T}, R_\mathrm{R}) / \min(R_\mathrm{T}, R_\mathrm{R}) \leq 1.50$; otherwise choose the larger ratio.

○ Transformer ○ RNN ○ Within 50%

(b) Which family benefits more? (12 points; 4 per row). Apply each row to both model families. Here N is the released RMSNorm (transformer) or LayerNorm (RNN). Tune each before/after model independently, using the same grid for every variant within its family:

All transformer variants	$\{.00075, .0015, .003, .006, .012\}$
All RNN variants	$\{.00025, .0005, .001, .002, .004\}$

	Before	After
(i)	No pre-norm: $x + F(x)$	Pre-norm: $x + F(N(x))$
(ii)	Pre-norm: $x + F(N(x))$	Post-norm: $N(x + F(x))$
(iii)	No RoPE	RoPE

For each family, compute $I = L^\star_{\text{before}} - L^\star_{\text{after}}$ for $L^\star$ as LR-tuned final validation loss. Choose **Transformer** if $I_\mathrm{T} - I_\mathrm{R} \geq .02$ (the transformer benefits more from the intervention), **RNN** if $I_\mathrm{R} - I_\mathrm{T} \geq .02$ (the RNN benefits more from the intervention), and **Similar** if $|I_\mathrm{T} - I_\mathrm{R}| < .02$ (both have similar benefits).

Q3. RNNs and transformers across time

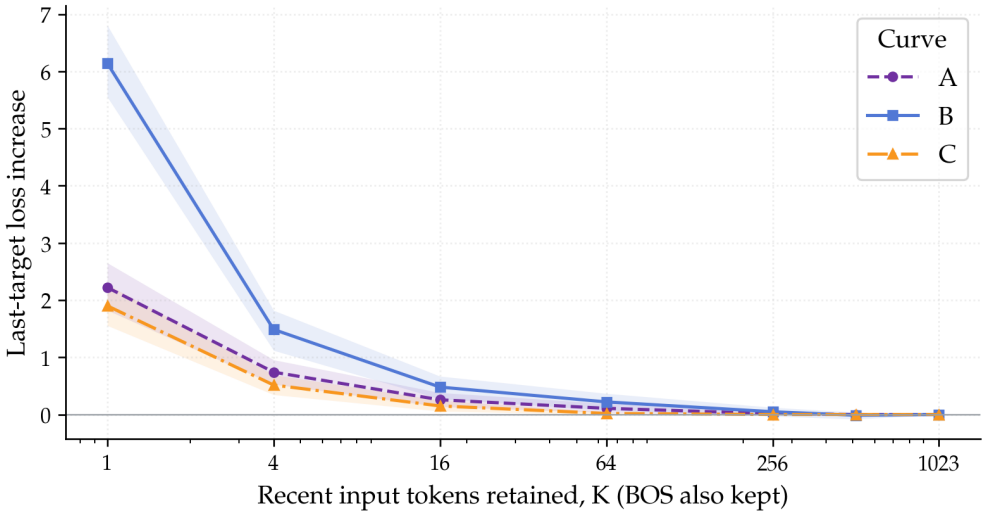
Difficulty: Easy

20 points

From A4 Problem 2. Compare the default transformer, Modern tanh baseline, and GRU.

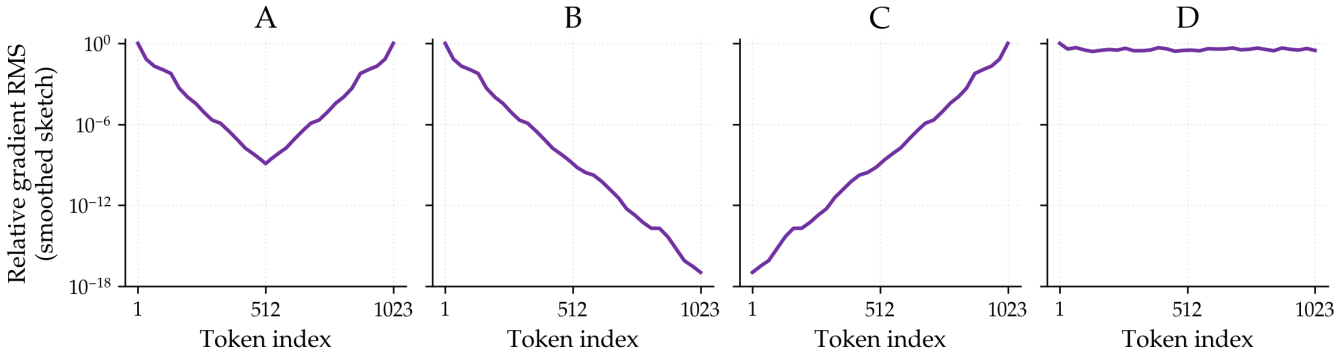
Diff against the A4 release: Appendix A.3, page 8.

(a) Context removal (12 points; 4 per curve). Match A–C to GRU, Modern tanh baseline, and transformer, using each once. For each target, retain the most recent K inputs and initial BOS, restart recurrent states, and preserve transformer positions. Score the same final target and subtract its loss with all 1,023 inputs retained. The curves average these paired loss increases over 128 targets. Moving left removes more context.



(b) Temporal gradients (8 points; 4 per row). For the Modern tanh baseline, match A–D to the last-token gradient $g_{\text{last}}(t) = \text{RMS}(\partial L_T / \partial h_{1,t})$ and full-loss gradient $g_{\text{full}}(t) = \text{RMS}(\partial \bar{L} / \partial h_{1,t})$, where $\bar{L} = T^{-1} \sum_{s=1}^{T} L_s$ with respect to the hidden state in layer 1. Choose exactly one plot per row, using two different letters; the other two plots are distractors.

Details: RMS averages over batch and hidden features. The panels are normalized, smoothed sketches where we divide each curve by its maximum, then interpolate in log space through 32 time positions. The y-axis is logarithmic.



Last-token loss gradient: $L = L_T$ ○ A ○ B ○ C ○ D

Full-loss gradient: $L = T^{-1} \sum_{s=1}^{T} L_s$ ○ A ○ B ○ C ○ D

Q4. Pinning residual-stream norms

Difficulty: Hard

10 points

From A4 Problems 1 and 3. Use the default 8-block transformer. Choose the final loss-regret interval $\Delta = L_{\text{changed}} - L_{\text{control}}$ for the intervention below.

Diff against the A4 release: Appendix A.4, page 9.

Pin stream norms halfway through training. At update 4,688 of the LR .003 baseline, fork two continuations. One continues unchanged. In the other, after each attention/MLP residual addition y, apply

$$y \leftarrow s_j \, y / \operatorname{RMS}(y).$$

where s_j is the mean tokenwise RMS on 16 validation sequences at the fork. Compare this constraint with the unchanged continuation at update 9,375. How much does the loss change after this modification? Choose the final loss-regret interval $\Delta = L_{\text{changed}} - L_{\text{control}}$.

Choices (loss regret):

A $\Delta < -.02$ (improves)

B $-.02 \leq \Delta < .02$ (little change)

C $.02 \leq \Delta < .05$

D $.05 \leq \Delta \leq .15$

E $\Delta > .15$ or all tried runs diverge

Q5. Residual designs across depth

Difficulty: Medium

22 points

From A4 Problems 1 and 3. Use the default 8-block transformer and released training recipe.

Diff against the A4 release: Appendix A.5, page 9.

(a) Keep only a whole-block skip (10 points). The default block has an attention skip and an MLP skip:

$$u = x + A(N_1(x)), \qquad y = u + M(N_2(u)),$$

where A is attention, M is the MLP, and N_1, N_2 are RMSNorms. Replace this rule by

$$y = x + M(N_2(A(N_1(x)))),$$

removing both internal skips and keeping one skip from the block input x to its output y. Train from scratch and tune over $\{.001, .003, .009\}$; compare with the separately tuned default transformer. Choose the final loss-regret interval $\Delta = L_{\text{changed}} - L_{\text{control}}$.

Choices (loss regret):

A $\Delta < -.02$ (improves)

B $-.02 \leq \Delta < .02$ (little change)

C $.02 \leq \Delta < .05$

D $.05 \leq \Delta \leq .15$

E $\Delta > .15$ or all tried runs diverge

(b) Read across depth (12 points). Separately train the full default attention-residual transformer at LR .006. Which strip best matches the final read router's attention over sources, immediately before the output norm/head?

Source 0 is the embedding; sources 1–16 are the raw attention/MLP outputs in depth order (odd: attention; even: MLP). Average over all 1,023 input positions of four held-out sequences. One strip is measured; three are constructed alternatives. Box values are rounded percentages, with a shared color scale.



○ A ○ B ○ C ○ D

Q6. Changing the recurrent update

Difficulty: Hard

20 points

From A4 Problem 4. Change the temporal update of the modern tanh baseline. Here x_t is the normalized input to a recurrent cell and h_t is its internal state carried to the next time step, before the depth MLP. Every layer starts with $h_0 = 0$ and has its own learned weights.

Diff against the A4 release: Appendix A.6, page 10.

(a) Rank four designs (12 points). Let $c_t = \tanh(Wx_t + Uh_{t-1} + b)$.

A:	Modern tanh baseline	$h_t = c_t,$
B:	Time residual	$h_t = h_{t-1} + c_t,$
C:	Normalized time residual	$s_t = h_{t-1} + c_t, \quad h_t = \dfrac{s_t}{\sqrt{\frac{1}{512}\sum_{j=1}^{512} s_{t,j}^2 + 10^{-5}}},$
D:	Full GRU	update below.

The full GRU update is

$$\begin{aligned} r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r), \\ z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z), \\ \tilde{h}_t &= \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \end{aligned}$$

Here σ is sigmoid and $\odot$ is elementwise multiplication.

Tune **all four designs** over the same LR grid $\{.001, .002, .004\}$. Let $L_A^\star, \ldots, L_D^\star$ be their best final validation losses. Rank lowest to highest loss, using >, <, and = (for losses within 0.01). Each correctly ordered pair earns 2 points.

(b) Fixed versus learned write gate (8 points). Keep the GRU's learned reset gate and candidate. Replace its write gate by a constant z, shared across all channels, layers, and time steps:

$$h_t = (1 - z)h_{t-1} + z\tilde{h}_t, \qquad h_0 = 0.$$

Train from scratch, tuning $z \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and LR $\eta \in \{.001, .002, .004\}$. Compare the best fixed-z model with the full GRU from (a), tuned over the same LR grid. Which interval contains the final validation loss gap

$$\Delta = \min_{z,\eta} L_{\text{fixed}}(z, \eta) - \min_{\eta} L_{\text{GRU}}(\eta)?$$

- ○ A: $\Delta \le 0$
- ○ B: $0 < \Delta \le .05$
- ○ C: $.05 < \Delta \le .10$
- ○ D: $.10 < \Delta \le .15$
- ○ E: $\Delta > .15$

Appendix A: Diffs against the A4 release

Each patch applies independently to assignments commit `4350166`. Q1–3 add diagnostic helpers; Q4–6 add model subclasses. Red lines are removed; green lines are added. All runs keep the released `training_config`, including linear decay and RMS logging.

A.1 — Q1: Complete-block norms

Select complete block outputs from each released variant's LR-selected checkpoint.

```
--- a/experiments/a4_stability/diagnostics.py
+++ b/experiments/a4_stability/diagnostics.py
@@ -141,2 +141,8 @@
     path.parent.mkdir(parents=True, exist_ok=True)
     path.write_text(json.dumps(clean(value), indent=2, allow_nan=False)+'\n')
+
+
+def block_norms(report, field):
+    import numpy as np
+    # Every block has two branches; omit embeddings and the readout norm.
+    return np.array([row[field] for row in report['full']['depth'][2::2]])
```

A.2 — Q2: Ratios and gains

Q2(a): default LRs .003/.001 (transformer/RNN). Q2(b): tune on the listed grids; compute before-minus-after loss.

```
--- a/experiments/a4_stability/diagnostics.py
+++ b/experiments/a4_stability/diagnostics.py
@@ -141,2 +141,9 @@
     path.parent.mkdir(parents=True, exist_ok=True)
     path.write_text(json.dumps(clean(value), indent=2, allow_nan=False)+'\n')
+
+
+def depth_ratio(first_block, last_block):
+    return last_block/first_block
+
+def loss_gain(before, after):
+    return before-after
```

A.3 — Q3: Context and both gradient objectives

Use the release's `context_loss(model, tokens, K, preserve_first=True)` on 128 paired targets. Its `profile` already computes both full-loss and last-token gradients while retaining the true internal recurrent states. Use the first four validation sequences from the Modern tanh baseline. For the displayed sketches, normalize each curve and interpolate in log space through 32 sampled time positions.

```
--- a/experiments/a4_stability/diagnostics.py
+++ b/experiments/a4_stability/diagnostics.py
@@ -141,2 +141,17 @@
     path.parent.mkdir(parents=True, exist_ok=True)
     path.write_text(json.dumps(clean(value), indent=2, allow_nan=False)+'\n')
+
+
+def context_increase(truncated, full):
+    import numpy as np
+    return np.asarray(truncated)-np.asarray(full)
+
+def temporal_sketch(report, objective):
+    import numpy as np
+    # The release profile already differentiates both objectives and retains h_t.
+    # Display a normalized sketch with log interpolation through 32 time positions.
+    g=np.array(report[objective]['hidden_gradient_rms'][0])
+    logg=np.log10(np.maximum(g/g.max(),1e-18))
+    knots=np.linspace(0,len(g)-1,32).astype(int)
+    smooth=10**np.interp(np.arange(len(g)),knots,logg[knots])
+    return smooth/smooth.max()
```

A.4 — Q4: Norm pinning

Fork update 4,688 including optimizer and RNG state. Measure 16 RMS values s_j in FP32 on the first 16 validation sequences, then fix these targets and rescale each residual output to its target RMS. All weights keep learning. Continue both branches to 9,375 with the same batches and remaining LR schedule.

```
--- a/experiments/a4_stability/p1_transformer_depth.py
+++ b/experiments/a4_stability/p1_transformer_depth.py
@@ -213,2 +213,19 @@
     from .sweeps import launch_one
     launch_one('d8')  # Edit the variant and LR, e.g. ('d8-no-residual', .001).
+
+
+class PinnedNormTransformer(Transformer):
+    """Freeze residual RMS targets; existing learned RMSNorm gains keep training."""
+    def __init__(self, config, *, targets=None, **kwargs):
+        super().__init__(config, **kwargs)
+        self.targets = None if targets is None else tuple(targets)
+        if self.targets is not None:
+            assert len(self.targets) == len(self.branches())
+
+    def run_depth(self, x, **context):
+        for j, branch in enumerate(self.branches()):
+            x = branch(x, **context)
+            if self.targets is not None:
+                scale = x.float().square().mean(-1, keepdim=True).clamp_min(1e-12).rsqrt()
+                x = x * (self.targets[j]*scale).to(x.dtype)
+        return x
```

A.5 — Q5: Residual designs

For (a), keep a skip around each complete attention–MLP composition, removing its internal skips. Train from scratch on the stated three-rate grid.

```
--- a/experiments/a4_stability/p1_transformer_depth.py
+++ b/experiments/a4_stability/p1_transformer_depth.py
@@ -213,2 +213,12 @@
     from .sweeps import launch_one
     launch_one('d8')  # Edit the variant and LR, e.g. ('d8-no-residual', .001).
+
+
+class BlockSkipTransformer(Transformer):
+    """Only the whole block has a skip: x + M(N2(A(N1(x))))."""
+    residual_type = NoResidual
+
+    def run_depth(self, x, **context):
+        for block in self.model.layers:
+            x = x + block(x, **context)
+        return x
```

For (b), use the released `AttentionTransformer`, with block size 1, at LR .006. Apply the release's `profile(model, tokens)` to four validation sequences; the helper selects the final read's mean routing weights. The model is unchanged. The three distractors are synthetic probability vectors.

```
--- a/experiments/a4_stability/diagnostics.py
+++ b/experiments/a4_stability/diagnostics.py
@@ -141,2 +141,8 @@
     path.parent.mkdir(parents=True, exist_ok=True)
     path.write_text(json.dumps(clean(value), indent=2, allow_nan=False)+'\n')
+
+
+def final_read_weights(report):
+    # Full 8-block AttnRes: embedding plus 16 raw attention/MLP outputs.
+    # The released profile already averages each router over batch and time.
+    return report['routing']['routers.16.softmax']
```

A.6 — Q6: Temporal update variants

A is the Modern tanh baseline (released `ModernRNN`); D is the released `GRU`. The subclasses below keep the modern depth stack and change only the recurrent update. The reset gate stays learned for every fixed write gate z. All designs in Q6(a) and Q6(b) use the same LR grid $\{.001, .002, .004\}$ and the release's linear schedule.

```
--- a/experiments/a4_stability/p4_recurrent_updates.py
+++ b/experiments/a4_stability/p4_recurrent_updates.py
@@ -61,2 +61,43 @@
     from .sweeps import launch_one
     launch_one('gru')  # Edit the variant and LR here.
+
+
+class ConstantWrite(nn.Module):
+    def __init__(self, value=1.):
+        super().__init__()
+        self.value = value
+
+    def project(self, x):
+        return x[..., :1]
+
+    def forward(self, projected, h):
+        return self.value
+
+class FixedWriteCell(GRUCell):
+    def make_update(self, config):
+        return ConstantWrite()
+
+class FixedWriteGRU(GRU):
+    """Learn the reset gate and candidate; fix the write gate z."""
+    cell_type = FixedWriteCell
+
+    def __init__(self, config, *, write=.5, **kwargs):
+        super().__init__(config, **kwargs)
+        assert 0 <= write <= 1
+        for block in self.stack:
+            block.recurrent.operation.cell.update.value = write
+
+class TimeResidualCell(TanhCell):
+    def step(self, projected, h):
+        return h + super().step(projected, h)
+
+class NormalizedTimeResidualCell(TimeResidualCell):
+    def step(self, projected, h):
+        y = super().step(projected, h)
+        return y * (y.float().square().mean(-1, keepdim=True)+1e-5).rsqrt().to(y.dtype)
+
+class TimeResidualRNN(ModernRNN):
+    cell_type = TimeResidualCell
+
+class NormalizedTimeResidualRNN(ModernRNN):
+    cell_type = NormalizedTimeResidualCell
```

Name _____ **SUNet ID** _____

your Stanford login, e.g. `jdoe`; not the 8-digit ID number

Write the matching letters or check a choice. Only these two pages are graded.

Q1. Norms across depth — 12 points

Variant	(a) Activation 6 pts; 2 each	(b) Gradient 6 pts; 2 each
No pre-norm	_____	_____
Post-norm	_____	_____
No residual	_____	_____

Assignment problem cited	Observations from the experiment	Your rationale
(a)	(a)	(a)
(b)	(b)	(b)

Q2. Across depth — 16 points

(a) Larger ratio: 4 pts ☐ Transformer ☐ RNN ☐ Within 50%

(b) Change 12 pts; 4/row	Transformer	RNN	Similar
(i) No pre-norm → pre-norm	☐	☐	☐
(ii) No post-norm → post-norm	☐	☐	☐
(iii) No RoPE → RoPE	☐	☐	☐

Assignment problem cited	Observations from the experiment	Your rationale
(a)	(a)	(a)
(b)	(b)	(b)

Q3. Across time — 20 points

(a) Context: 12 pts; 4/curve GRU _____ Modern tanh baseline _____ Transformer _____

(b) Pick one plot per row 8 pts; 4/row	A	B	C	D
Last-token loss gradient	☐	☐	☐	☐
Full-loss gradient	☐	☐	☐	☐

Assignment problem cited	Observations from the experiment	Your rationale
(a)	(a)	(a)
(b)	(b)	(b)

Name _____ **SUNet ID** _____

your Stanford login, e.g. `jdoe`; not the 8-digit ID number

Q4. Pinning residual-stream norms 10 points

	A	B	C	D	E
Fix stream RMS halfway	☐	☐	☐	☐	☐

Assignment problem cited	Observations from the experiment	Your rationale

Q5. Residual designs across depth 22 points

	A	B	C	D	E
(a) Only a whole-block skip 10 pts	☐	☐	☐	☐	☐

(b) Routing pattern: 12 pts ☐A ☐B ☐C ☐D

Assignment problem cited	Observations from the experiment	Your rationale
(a)	(a)	(a)
(b)	(b)	(b)

Q6. Changing the recurrent update 20 points

(a) Rank: 12 pts; 2/pair ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

(b) Loss gap: 8 pts ☐A ☐B ☐C ☐D ☐E

Assignment problem cited	Observations from the experiment	Your rationale
(a)	(a)	(a)
(b)	(b)	(b)