1.

Expected: 0.1973753273487091, got: 0.1973753273487091, max error: 0.0

Max error all\_h: 4.999339580535889e-05

Max error last\_h: 2.498924732208252e-05

(b)

Max error all\_h: 4.699826240539551e-05

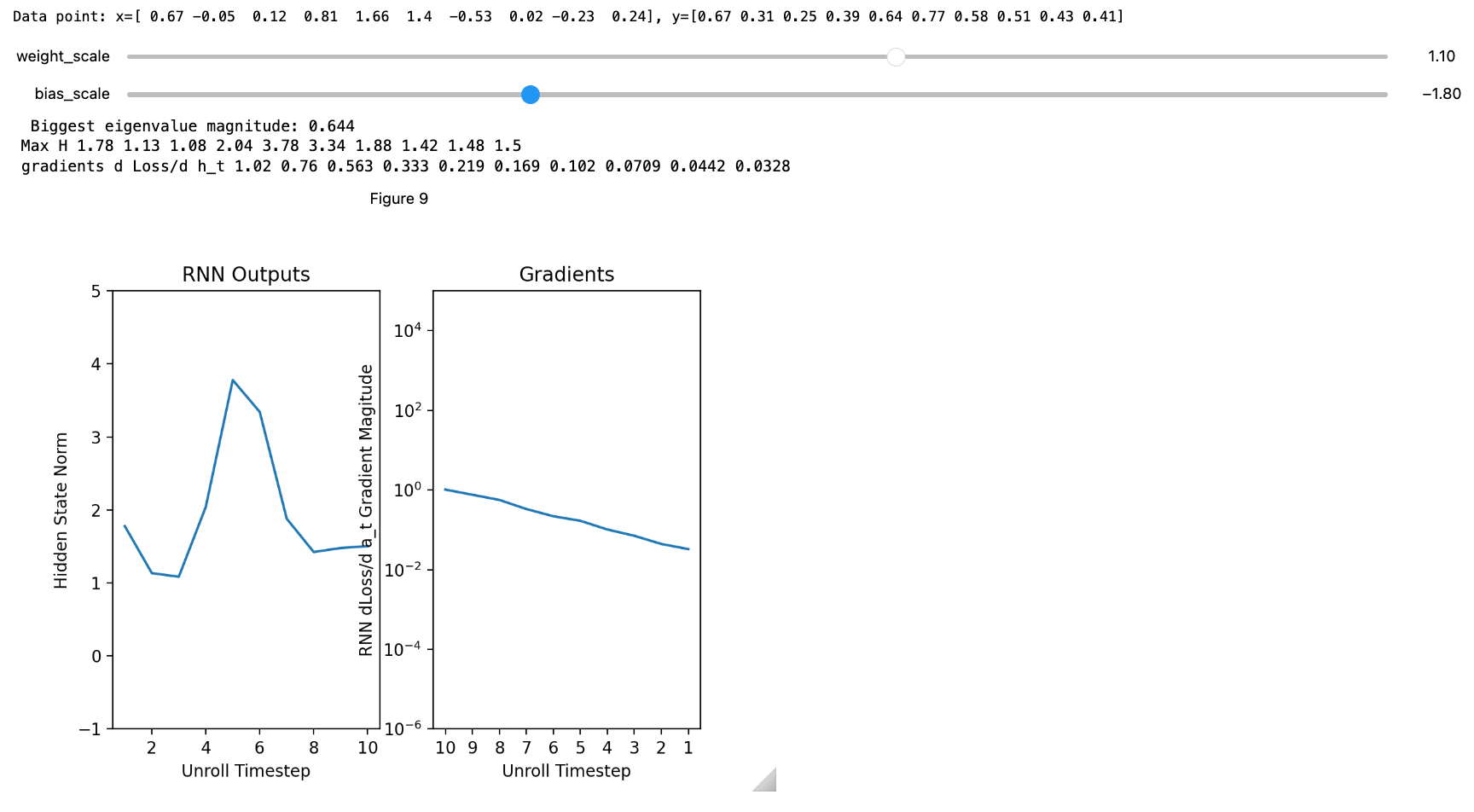
Max error last\_h: 4.3138861656188965e-05

(c)

Max error loss\_all: 3.314018249511719e-05

Max error loss\_last: 2.384185791015625e-07

(d)



(e)

With loss on the last step only, the backpropagated Jacobian at each step is Wᵀ,, so gradients contain powers of W.

* If the largest singular value σmax(W) < 1, gradients vanish exponentially with distance from the last step.
* If σmax(W) > 1, gradients explode exponentially.
* If σmax(W) ≈ 1, gradients stay roughly constant.

(f)

More vanishing: tanh

Hidden states are bounded in [-1, 1] and saturate; derivatives are ≤ 1 and typically much < 1 away from 0

More exploding: ReLU

Active paths keep multiplying by W with no saturating shrinkage

(g)

With last\_target\_only = False, you add a loss at every time step. This injects gradient at each time, so earlier steps don’t rely on long chains of W from the last step, and thus vanishing is much less severe.

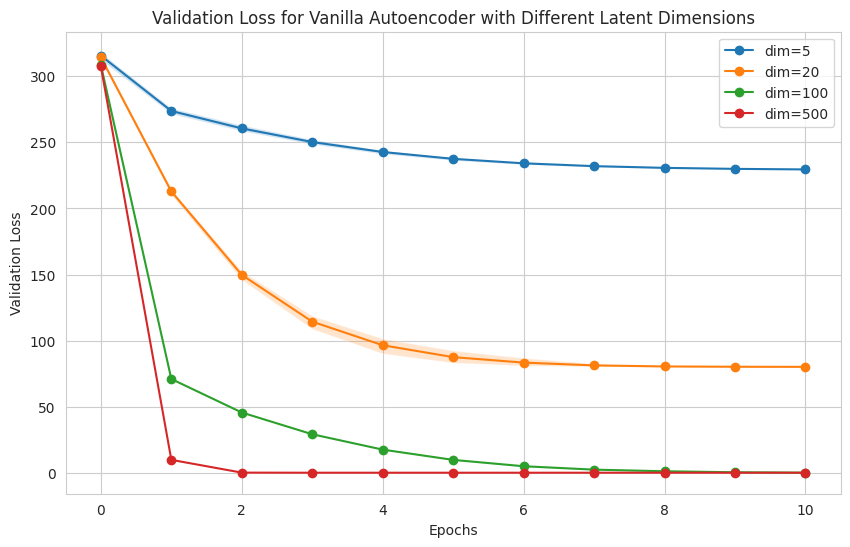
* When only the last step has a loss, gradients to early timesteps must pass through many recurrent multiplications, shrinking or blowing up exponentially.
* When every step has a loss, each step’s parameters receive a “local” gradient that traveled only a few steps, so those gradients are larger and more stable.
* But if a target at time T truly depends on information from a far earlier time t, the gradient for that dependency still traverses T−t steps and will still vanish/explode the same way.

1. The discrimination and bias in decision-making in employment or education.

3.

(a)

(i)

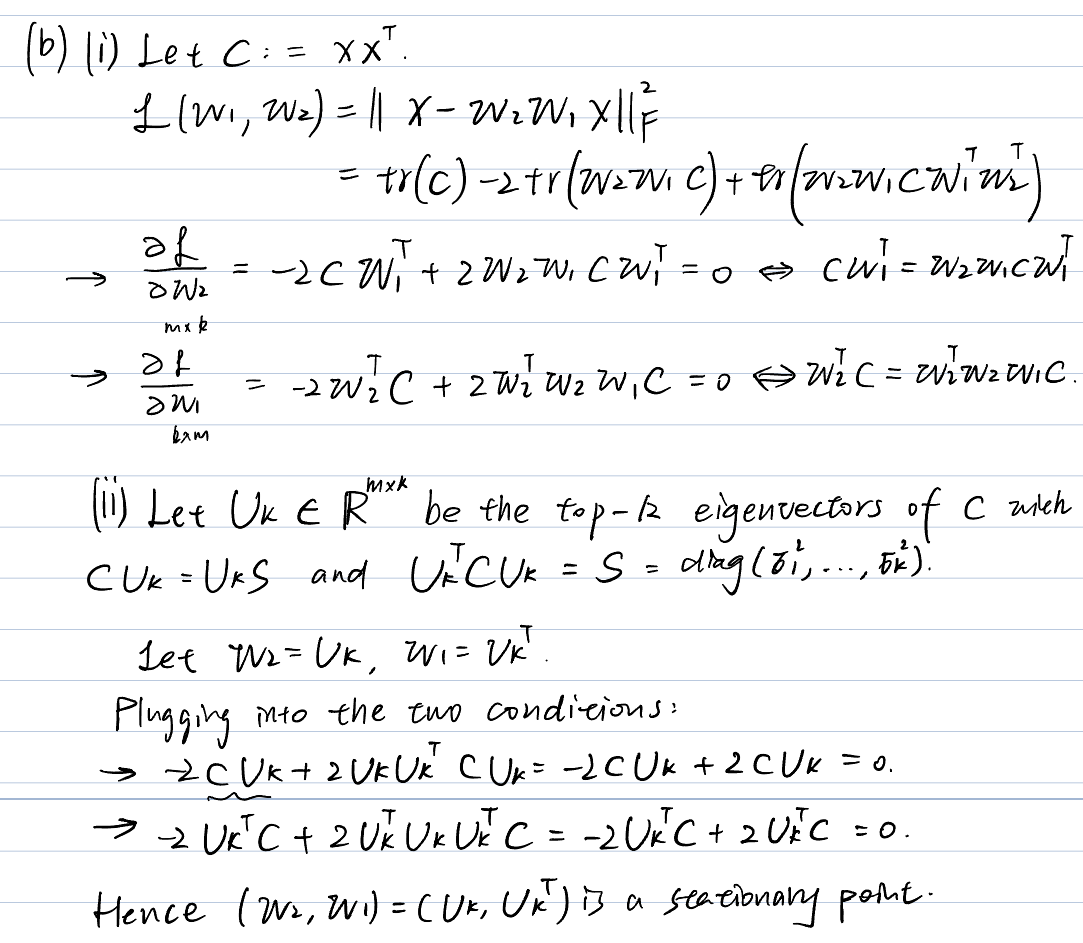
(ii)

* Reconstruction: Increases monotonically with larger latent size and then plateaus; the elbow appears around the intrinsic dimensionality. It improves rapidly up to ≈20 and shows diminishing returns beyond that.
* Linear probe accuracy: Rises sharply as latent size grows from very small to around ≈20, then plateaus or can dip slightly beyond that.

Why:

* The data are 100D with roughly 20 high‑variance, label‑relevant dimensions.
* With a very small bottleneck, the autoencoder can’t capture all salient structure → higher reconstruction error and poor linear separability.
* As latent size approaches the intrinsic task‑relevant dimension (~20), the model can encode the key factors, and thus reconstruction error drops and linear separability peaks.
* Increasing latent size past ~20 mostly captures low‑variance/nuisance variation that doesn’t help classification (and can dilute separability), so reconstruction keeps improving slightly

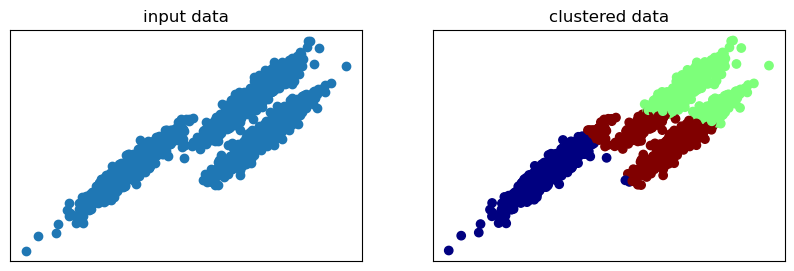
(b)



4.

5.

(a)



* The K-means algorithm did not work effectively for this dataset. Looking at the clustered data, K-means has divided the data into three clusters, but these clusters don't properly capture the natural structure of the data(three elongated, curved branches).
* K-means assumes clusters are spherical and similar in size, uses Euclidean distance as its similarity metric.
* In this dataset, points that belong to the same natural cluster might be far apart in Euclidean distance.

(f)

Observation: After normalizing the three feature vectors per point, the scatter shows the capture the natural structure of the data(three elongated, curved branches).

Why normalization works:

* Normalizing each vector to unit L2-norm makes similarity depend on direction (cosine) rather than magnitude. On the unit sphere, Euclidean distance is monotonic with 1 − cosine similarity, so k-means groups by angle. This removes scale/degree effects that previously pulled points with larger norms together even if their directions differed.

(g)

