A Effect of multidimensional output on learned representations

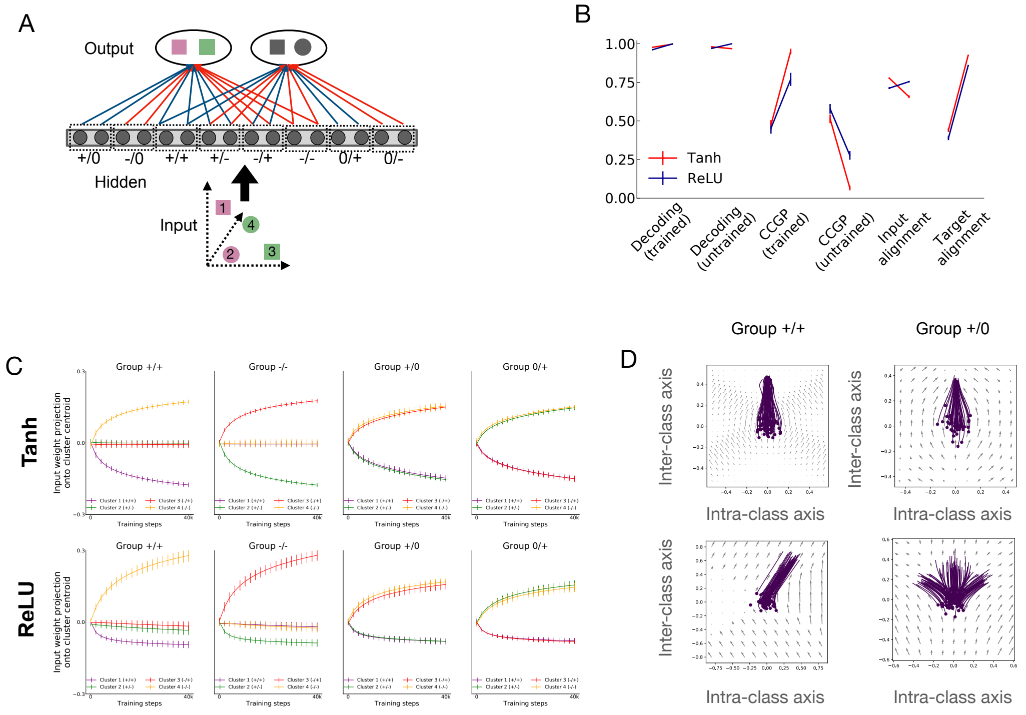


Figure 1: A. Schematic of binary classification task with unstructured inputs and two outputs. The outputs are trained to classify four input clusters according to two different labelings of the inputs. In this case the hidden layer of the network is divided into eight groups of neurons defined by their two output weights (positive: blue, negative: red, zero: no line). B. Measures of representational geometry at network initialization (left of each line segment) and following training (right of each line segment). Error bars indicate standard deviation over 20 simulated networks. D. Alignment (dot product similarity) of input weights to hidden layer neurons with the four input cluster centers. Shown for four of the eight hidden layer neuron groups. Error bars indicate standard deviation across neurons within a single network. E. Trajectories of input weights to hidden layer neurons along the inter-class ($x_1 + x2 - x_3 - x_4$) and the intra-class axis ($x_1 - x_2$). Each line sigment represents an individual neuron from a simulation, and small circles indicate the beginning of the trajectory (at network initialization). Shown for hidden layer neurons with an output weight of $+1$. Vector field background indicates the expected average direction that weights will evolve due to gradient of the task objective.

Here we include an in-depth exploration of networks trained with two targets and four items. We modified our tasks by now assigning each of the four input clusters to two different binary labels: (1, 1), (1, 0), (0, 1), and (0, 0), respectively (Fig 1). We modified the network architecture by adding a second output unit; each output unit was tasked with predicting one of the two labels. Thus, in this task, there were two trained dichotomies (x_1/x_2 vs. x_3/x_4, and x_1/x_3 vs. x_1/x_4) and one untrained dichotomy (x_1/x_4 vs. x_2/x_3). For analysis, we again used frozen readout weights discretized into eight groups (Fig 1A), which we found was sufficient to approximate the behavior of networks trained with randomly initialized output weights. Again, we found a noticeable difference in the learned representations of ReLU and Tanh networks, with Tanh networks reflecting the geometry of the targets more than ReLU networks (Fig 1B). Again, we analyzed the source of these phenomena by tracking the alignment of input weights to each neuron with the four input cluster centroids. In general, we observed that neurons grew positively tuned to inputs which matched their output weights, negatively tuned to inputs mismatched with their output weights, and not tuned at all to inputs matched with one output weight and mismatched with the other (Fig. 1C). However, in the ReLU networks, a clear asymmetry emerged in the degree of positive tuning to matched input clusters and the degree of negative tuning to mismatched input clusters. As a result, in the ReLU networks, roughly half the neurons (those with nonzero outputs to both class readout units) developed selectivity primarily for one of the four input clusters. As a result, these ReLU units developed strong selectivity for intra-class differences, while Tanh units did not (Fig. 1D). This behavior was observed to be uniform across the hidden neurons (Fig. 1D).

Thus, both the multi-output task produced similar results as the similar output task in terms of representational geometry, the source of these effects was somewhat different. In both cases, the gradient of the task objective drives input weights to accrue inter-class selectivity in Tanh networks. However, for ReLU networks, the picture is different. In the single-output case, the emergence of intra-class selectivity arises due to heterogeneity in initial conditions of input weights, while in the multi-output case it also arises due to heterogeneity in initial conditions of output weights.

B Impact of constraints on the output weights

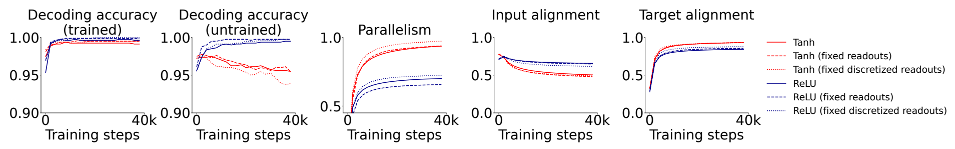


Figure 2: Measures of representational geometry during training for three cases: randomly initialized trainable readouts, randomly intiialized fixed readouts, and discretized (binary ± 1) fixed readouts.

In our experiments we used frozen output weights with discretized values. In Fig. 2 we show that the behavior of representation learning is essentially the same when we relax the discretization and allow the readouts to be trained.

C Further details on random aligned kernel sampling

Our sampling procedure for tasks with a specified input-output alignment used in Section 5 has some important considerations, which will be further elaborated in this section. Firstly, obviously, the sampled matrices must be symmetric and positive semi-definite (SPSD), and they need to have the required correlation value. Secondly, kernels with the same alignment value can have different dimensionality (i.e. eigenspectra).

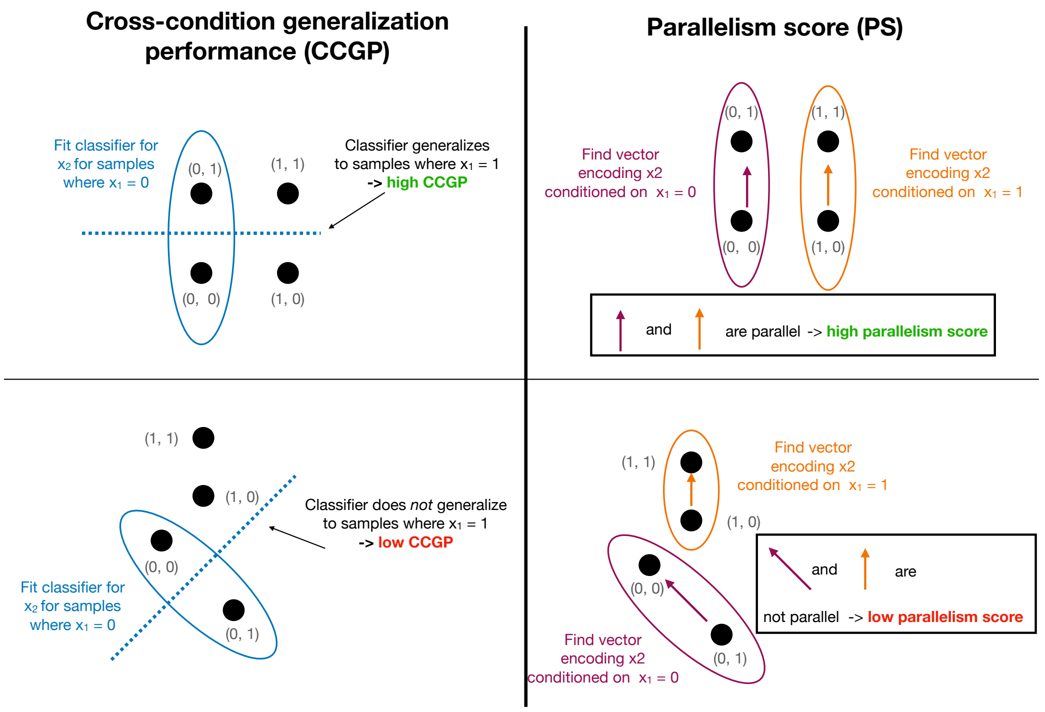
It is worth clarifying that we assume all kernels are already centered, meaning that the features used to compute them have been mean-subtracted. Or, equivalently, that their nullspace contains the vector $\mathbf{1}$ of all ones. They should also have unit trace, $\mathrm{Tr}K = 1$. Both are non-restrictive assumptions, since the CKA is invariant to scaling and shifting of features.

The set of matrices with a given correlation value, c, to a reference matrix K_Y is the solution set of a quadratic form[1], while the SPSD matrices (of unit trace) are a compact convex body. We will refer to this set as $\mathcal{K}_c$. Our procedure is to randomly sample a SPSD matrix, and project it onto the quadric surface, taking care to remain SPSD. This method certainly does not ensure uniformity, but it is simple and empirically shows a wide spread of samples even in relatively high dimensions.

An important consideration with this method is the linear dimensionality of the input geometry, intuitively a measure of correlations between input patterns. If we define dimensionality in terms of the participation ratio of the eigenvalues, λ, of K:

$$\mathrm{p.r.}(K) = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2}$$

[1] $C(K_X, K_Y) = c \Leftrightarrow \mathrm{Tr}(K_X K_Y)^2 = c^2 \mathrm{Tr}(K_X K_X)\mathrm{Tr}(K_Y K_Y)$.



Cross-condition generalization performance (CCGP)
Parallelism score (PS)
Fit classifier for x₂ for samples where x₁ = 0
(0, 1)
(1, 1)
(0, 0)
(1, 0)
Classifier generalizes to samples where x₁ = 1 -> high CCGP
Find vector encoding x2 conditioned on x₁ = 0
Find vector encoding x2 conditioned on x₁ = 1
and
are parallel -> high parallelism score
Classifier does not generalize to samples where x₁ = 1 -> low CCGP
not parallel -> low parallelism score

Figure 3: Examples of representational geometries with high and low CCGP and PS, for data generated by two underlying binary variables, i.e. data points of the form (x_1, x_2) where $x_i \in \{0, 1\}$.

then, per our assumption of unit trace, this is just $\|K\|_F^{-2}$. In the special case when $c = 0$ the set $\mathcal{K}_0$ is convex, and this implies that there is a unique kernel matrix $K_{\max}$ which maximises $\mathrm{p.r.}(K)$, i.e. minimises $\|K\|_F$. If we draw a line between K_Y and $K_{\max}$, we get the set of input kernels which form the solid lines of Fig. 4B and Fig. 5 of the main text.

D Explanation of CCGP and PS metrics

See Figure 3.

E Comparison of weight dynamics at different noise levels in ReLU networks

In Section 5.2 we observed in ReLU networks an interesting increase in the values of target alignment / parallelism / CCGP, and decrease in the value of input alignnment, for intermediate values of the input noise σ. Here (Fig. 4) we provide an analysis that sheds some light on this phenomenon. For diffeerent noise values, we tracked the relationship between the value of the input weights to a network hidden-layer neuron and the corresponding value following training. In the absence of noise, neurons initialized with sufficient intra-class selectivity relative to their initial inter-class selectivity are destined to maintain it over time (remain in the purple or green regions in Fig. 4 – see weight trajectory plots in Fig. 1D, 2C for an explanation of why). In the presence of noise, neurons with substantial intra-class vs. inter-class selectivity at initialization can nevertheless evolve to develop primarily inter-class selectivity over the course of training. Note that this explanation does not account for why, at extremely high noise values, target alignment drops again (Fig. 3C) – we leave an in-depth characterization of this phenomenon to future work, but suspect it arises because the input noise grows substantial enough for the weights to accrue significant projections along axes other than the two we visualize here.

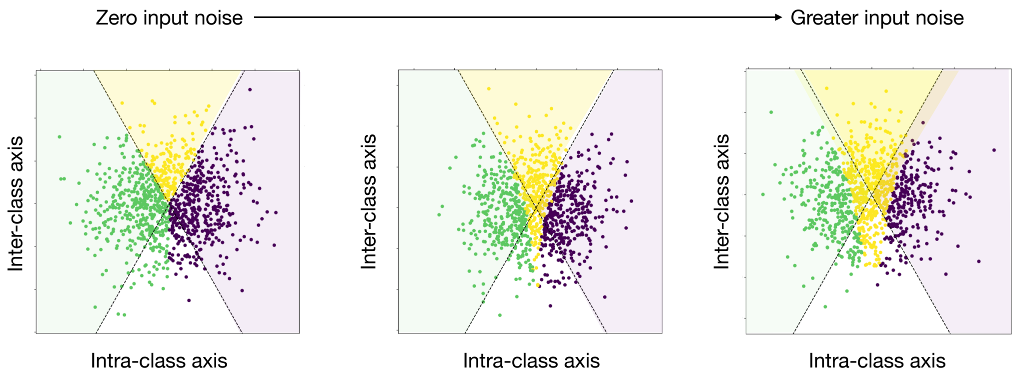


Figure 4: Effect of input noise on training dynamics of ReLU networks trained on the tasks in Section 5.2, in the case of maximal separability of the trained dichotomy, varying the level of input noise σ^2 during training. Each point corresponds a hidden-layer neuron in the network, and the position of the point indicates the value of the input weights for that neuron at initialization. The colors of the dots indicate which of the three colored regions the input weights end up in at the end of training.

F Explanation of tasks used in synthetic multi-layer network experiments

Here we describe the tasks used in Section 7. Formally, the "hard" task is generated by sampling input data of minimal possible dimensionality (5), or equivalently an input data kernel of rank 5, subject to the constraint of zero input-output kernel alignment. Intuitively, this task involves inputs generated by 5 binary latent variables, and the input-output function requires a nonlinear conjunction of all 5 (e.g. whether the number of latent variables set to 1 is even or odd).

The "easy" task involves sampling inputs of maximal possible dimensionality (27 dimensions, i.e. a rank-27 input data kernel) subject to the kernel alignment constraint. Intuitively, this task, the targets require nonlinear conjunction of only two of the input dimensions.