# The Effect of Model Capacity on the Emergence of In-Context Learning

**Anonymous authors**
Paper under double-blind review

## Abstract

This paper investigates the relationship between model capacity and the emergence of in-context learning under a simplified statistical framework in the transformer model. When model capacity is restricted enough, transformers shift from learning the Bayes optimal estimator for the training task distribution to an estimator that is suitable for out-of-distribution tasks. This shift is attributed to the restricted model's inability to fully memorize the training task distribution. Further experiments examine how the transformer's hyper-parameters impact its capacity for memorization.

## 1 Introduction

In recent years, transformers [15] have emerged as a foundational architecture in deep learning by demonstrating remarkable performance across many domains [5, 11, 16, 18]. A notable capability of these models is *in-context learning* [4], whereby trained transformers learn new decision rules at inference time when presented with a concatenation of labeled training samples and unlabeled test samples as each sequential input.

To better understand the mechanics of in-context learning, a substantial body of research observes the ability of transformers to perform linear regression in-context by training models to solve a synthetic task where inputs are labeled using a randomly drawn linear predictor [1, 6, 17]. Further work [12] considers a particular multi-task setting where the weight vectors of the planted linear predictor are randomly sampled from a finite set of size $k$. The authors observe a phase transition as $k$ increases from a Bayes estimator (which performs optimally on the planted tasks) to a ridge regression estimator (which sacrifices performance on the $k$ planted tasks for better out-of-distribution performance). They hypothesize that this contrast between Bayesian and "non-Bayesian" in-context learning is related to the limited memorization capabilities of transformers that may not be resolved by increasing model capacity.

Here, we examine in-context learning using a simpler statistical estimation problem—the normal means problem—and study the effect of model capacity in greater detail. The normal means problem captures fixed design linear regression in a Gaussian model under an orthonormal design, which avoids the extra variability that comes from a random design. We empirically analyze the analogous phase transition between the Bayesian "memorization" regime and the non-Bayesian regime, where the learned estimator resembles a shrinkage estimator in the latter. This analysis reveals that the location of this phase transition of the learned in-context learning algorithm is a function of multiple notions of model capacity, including depth, width, and number of training epochs.

**Related work on memorization.** Beyond the relevance of our experiments to in-context learning, we complement existing literature on the memorization capabilities of neural networks [3, 10, 14, 13, 19] by focusing our analysis on transformer architectures. While recent work [8] evaluates transformer memorization abilities as a function of width, we consider a wider range of model parameterizations. We also study a different sense of memorization in which observations are noisy, and the inference task is one of denoising and has connections to in-context learning.

## 2 Preliminaries

### 2.1 Normal Means Problem

In the normal means problem, the observation is a random vector $\mathbf{x} \in \mathbb{R}^n$ sampled from a multivariate normal distribution $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_n)$, where $\sigma^2 > 0$ is the known variance of each coordinate of $\mathbf{x}$, and $\boldsymbol{\mu} \in \mathbb{R}^n$ is the unknown mean of $\mathbf{x}$.[1] The statistician's goal is to construct an estimator $\hat{\boldsymbol{\mu}} = \hat{\boldsymbol{\mu}}(\mathbf{x})$ for $\boldsymbol{\mu}$ based on the observation $\mathbf{x}$. We evaluate $\hat{\boldsymbol{\mu}}$ with the squared Euclidean distance loss: $L(\hat{\boldsymbol{\mu}}, \boldsymbol{\mu}) = \|\hat{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2^2$. We regard the pair $(\mathbf{x}, \boldsymbol{\mu})$ as an *instance* of the normal means problem.

The normal means problem captures a fixed design linear regression problem with orthonormal design. Namely, let $A \in \mathbb{R}^{m \times n}$ be the design matrix with orthonormal columns, and let $\mathbf{y} = A\boldsymbol{\beta} + \boldsymbol{\epsilon}$ be the response vector, where $\boldsymbol{\beta}$ is a vector of regression coefficients and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I_m)$ is a vector of normally distributed errors. Then $\mathbf{x} := A^{\intercal}\mathbf{y} = \boldsymbol{\beta} + \boldsymbol{\epsilon}'$, where $\boldsymbol{\epsilon}' = A^{\intercal}\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I_n)$ is the observation in an instance of the normal means problem with mean $\boldsymbol{\mu} := \boldsymbol{\beta}$.

### 2.2 Training and Bayes Estimators

We investigate how transformers can be trained to produce estimates in the normal means problem. During training, the weights $W$ of a transformer $f_W : \mathbb{R}^n \to \mathbb{R}^n$ are updated iteratively with a gradient-based optimization algorithm that aims to minimize the *empirical risk*

$$\frac{1}{T}\sum_{t=1}^{T} L(f_W(\mathbf{x}_t), \boldsymbol{\mu}_t) \tag{1}$$

on several instances $(\mathbf{x}_1, \boldsymbol{\mu}_1), \dots, (\mathbf{x}_T, \boldsymbol{\mu}_T)$ of the normal means problem. The planted means $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_T$ are sampled i.i.d. from a training distribution $\pi$. Conditioned on each $\boldsymbol{\mu}_t$, each corresponding $\mathbf{x}_t$ is sampled independently as $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \sigma^2 I_n)$ according to the normal means problem above. Since an instance of the normal means problem corresponds to a fixed design linear regression problem, the evaluation of the transformer $f_W$ on $\mathbf{x}$ can be regarded as performing (an assisted form of) in-context linear regression.

In our experiments, we train with a discrete distribution $\pi = \pi_S$ that is uniform on a finite set $S \subset \mathbb{R}^n$ of cardinality $k := |S|$. The set $S$ consists of $k$ vectors drawn (once and for all) independently from $\mathcal{N}(0, I_n)$; this way of generating the vectors is used to avoid unusual coincidences among the vectors.

We regard the transformer $f_W$ trained to minimize Eq. (1) as an estimator of the planted mean, $\hat{\boldsymbol{\mu}}(\mathbf{x}) = f_W(\mathbf{x})$. Our experiments investigate the abilities of trained transformers of different model capacities to solve the normal means problem and perform memorization by measuring the similarity of $\hat{\boldsymbol{\mu}}$ to two fixed estimators that depend on the distribution $\pi$.

The training setup described above is reminiscent of the Bayesian problem in which $\pi$ is regarded as a prior distribution for $\boldsymbol{\mu}$, and an estimator $\hat{\boldsymbol{\mu}}$ is evaluated according to its *Bayes risk* [9]

$$\mathbb{E}_{\boldsymbol{\mu}}\mathbb{E}_{\mathbf{x}|\boldsymbol{\mu}} L(\hat{\boldsymbol{\mu}}(\mathbf{x}), \boldsymbol{\mu}), \tag{2}$$

and the empirical risk in Eq. (1) is a finite-sample estimate of the Bayes risk. Due to our choice of the squared Euclidean distance as the loss function, the estimator that minimizes the Bayes risk is the posterior mean function $\mathbf{x} \mapsto \mathbb{E}_{\boldsymbol{\mu}}[\boldsymbol{\mu} \mid \mathbf{x}]$.

When $\pi = \pi_S$, the posterior mean function is the *memorization estimator* $\hat{\boldsymbol{\mu}}_{\text{mem}}$, which makes the maximally informed guess of $\boldsymbol{\mu} \in S$ based on $\mathbf{x}$:

$$\hat{\boldsymbol{\mu}}_{\text{mem}}(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\mu} \sim \pi_S}[\boldsymbol{\mu} \mid \mathbf{x}] = \sum_{\boldsymbol{\mu} \in S} \frac{\phi_{\boldsymbol{\mu}, \sigma^2 I_n}(\mathbf{x})}{\sum_{\boldsymbol{\mu}' \in S} \phi_{\boldsymbol{\mu}', \sigma^2 I_n}(\mathbf{x})} \boldsymbol{\mu}, \tag{3}$$

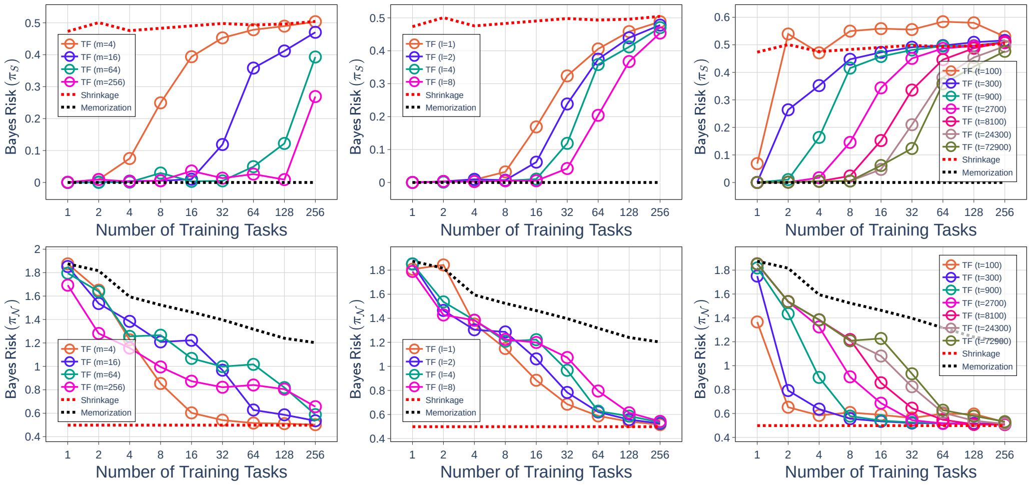
where $\phi_{\boldsymbol{\mu}, \sigma^2 I_n}$ is the density function for $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_n)$.

When $\pi = \pi_{\mathcal{N}} := \mathcal{N}(0, I_n)$, the posterior mean function is the *shrinkage estimator* $\hat{\boldsymbol{\mu}}_{\text{shr}}$, which does not have access to the set $S$:

$$\hat{\boldsymbol{\mu}}_{\text{shr}}(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\mu} \sim \pi_{\mathcal{N}}}[\boldsymbol{\mu} \mid \mathbf{x}] = \frac{1}{1 + \sigma^2}\mathbf{x}. \tag{4}$$

This estimator shrinks $\mathbf{x}$ towards the origin as a function of the fixed variance $\sigma^2$.

[1] In Appendix C, we also consider a setting in which $\sigma^2$ is unknown.



(a) Varying Hidden Dimension (b) Varying Depth (c) Varying # of Training Epochs

Figure 1: Effect of varying hidden dimension size, depth, and number of training epochs on the learned estimator. Models are evaluated on both in-distribution (top) and out-of-distribution (bottom) tasks. By default, we use $(m, l, t) = (16, 4, 100000)$.

## 3 Experiments

We train transformers using instances sampled from the training distribution based on $\pi_S$. Transformers are subsequently evaluated in-distribution according to their Bayes risk based on $\pi = \pi_S$, as well as out-of-distribution according to their Bayes risk based on $\pi = \pi_{\mathcal{N}}$. We examine the effects of individually increasing the size of the hidden dimension $m$, number of layers $l$, and the number of training epochs $t$ (equivalent to the amount of training data) on the estimators learned by transformers. Experimental details are included in Appendix B.

**Evaluation with In-Distribution Tasks.** Observe that increasing the model's capacity across all dimensions lead to an increase in the task diversity where the transformer's risk aligns with the memorization estimator (Figures 1a, 1b). However, it is noteworthy that for fixed model depth and training time, the benefits of scaling the hidden dimension diminishes.

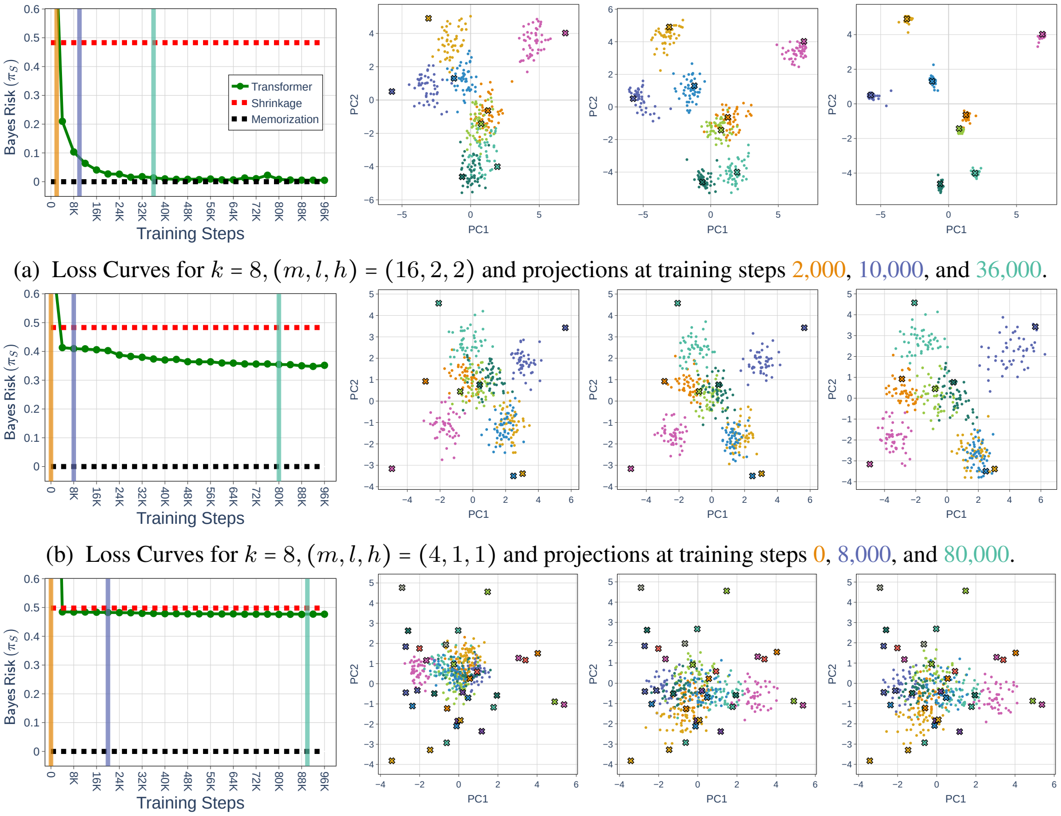
Our results are in line with previous findings of Raventós et al. [12], who reported a similar shift in the type of estimator learned by transformers based on task diversity. Furthermore, with sufficient model capacity, increasing the amount of training data also enables the transformer to match the memorization estimator for a larger task diversity (Figure 1c).

**Properties of Intermediate Representations.** While many of the transformer estimators with risks plotted in Figure 1 coincide almost exactly with $\hat{\boldsymbol{\mu}}_{\text{mem}}$ or $\hat{\boldsymbol{\mu}}_{\text{shr}}$, other risk curves indicate a lack of convergence to either estimator. Figure 2 visualizes the behaviors of trained transformer estimators throughout their training process in order to show their rate of convergence to $\hat{\boldsymbol{\mu}}_{\text{mem}}$ and low-dimensional representations of their predictions over several iterations.

To understand how the transformer memorizes the tasks $S$, we examine the geometric transition of the transformer's estimates across training steps. In Figure 2, we visualize the principal component analysis (PCA) projections of the transformer's estimates $\hat{\boldsymbol{\mu}}(\mathbf{x})$ relative to their planted means.

These plots enable an analysis of the behavior of "in-between" estimators $\hat{\boldsymbol{\mu}}$ (cf. Figure 2b) that converge to neither the shrinkage estimator $\hat{\boldsymbol{\mu}}_{\text{shr}}$ nor the memorization estimator $\hat{\boldsymbol{\mu}}_{\text{mem}}$. In some (but not all) cases, rather than memorizing a small subset of $\boldsymbol{\mu}$ vectors, these estimators interpolate between the two extremes by being simultaneously biased towards their corresponding $\boldsymbol{\mu}$ vectors and shrunk towards the origin (Figure 2c).

When the transformer has sufficient model capacity to converge to the memorization estimator, the tasks $\boldsymbol{\mu} \in S$ are not memorized one by one. Instead, the transformer first learns an in-between estima-



(a) Loss Curves for $k = 8, (m, l, h) = (16, 2, 2)$ and projections at training steps 2,000, 10,000, and 36,000.

(b) Loss Curves for $k = 8, (m, l, h) = (4, 1, 1)$ and projections at training steps 0, 8,000, and 80,000.

(c) Loss Curves for $k = 32, (m, l, h) = (4, 1, 1)$ and projections at training steps 0, 20,000, and 90,000.

Figure 2: PCA projections of transformer estimates for various task diversities and model capacities. Each $\boldsymbol{\mu} \in S$ vector is represented with a colored cross. Transformer estimates for each $\boldsymbol{\mu}$ are colored accordingly.

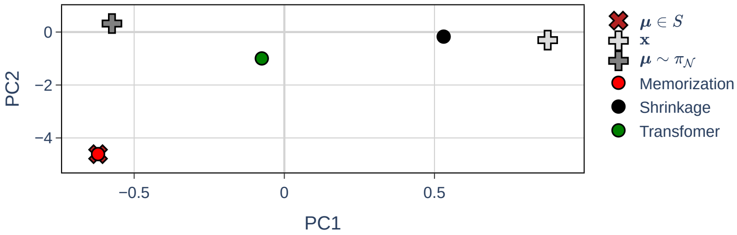


Figure 3: PCA projections of estimator predictions for an out-of-distribution sample $\mathbf{x}$ with $\boldsymbol{\mu} \sim \pi_{\mathcal{N}}$. We use the transformer model with parameters $(m, l, h) = (16, 2, 2)$ trained on $k = 8$ tasks.

tor (Figure 2a) and transitions towards uniformly producing tight estimates around the corresponding $\boldsymbol{\mu}$ vectors.[2]

Furthermore, we note an observation where transformers that align with $\hat{\boldsymbol{\mu}}_{\text{mem}}$ in-distribution (cf. Figure 2a) behave differently on out-of-distribution tasks. Figure 3 plots the predictions for an out-of-distribution test sample $\mathbf{x}$. Observe that the transformer's prediction $\hat{\boldsymbol{\mu}}(\mathbf{x})$ resides in an intermediate position relative to $\hat{\boldsymbol{\mu}}_{\text{mem}}(x)$ and $\hat{\boldsymbol{\mu}}_{\text{shr}}(x)$. This suggests that the transformer learns a robust estimator that performs optimally in-distribution and moderately on out-of-distribution tasks.

[2] For a comprehensive transition of the projections, see Appendix D.2.

## REFERENCES

[1] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models, 2023.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[3] Eric B Baum. On the capabilities of multilayer perceptrons. *Journal of Complexity*, 4(3):193–215, 1988. ISSN 0885-064X. doi:https://doi.org/10.1016/0885-064X(88)90020-9. URL `https://www.sciencedirect.com/science/article/pii/0885064X88900209`.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[6] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 2023.

[7] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[8] Junghwan Kim, Michelle Kim, and Barzan Mozafari. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=8JCg5xJCTPR`.

[9] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer New York, NY, 1998.

[10] Sejun Park, Jaeho Lee, Chulhee Yun, and Jinwoo Shin. Provable memorization via deep neural networks using sub-linear parameters. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 3627–3661. PMLR, 15–19 Aug 2021. URL `https://proceedings.mlr.press/v134/park21a.html`.

[11] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[12] Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression, 2023.

[13] Eduardo D. Sontag. Shattering All Sets of 'k' Points in "General Position" Requires $(k-1)/2$ Parameters. *Neural Computation*, 9(2):337–348, 02 1997. ISSN 0899-7667. doi:10.1162/neco.1997.9.2.337. URL `https://doi.org/10.1162/neco.1997.9.2.337`.

[14] Gal Vardi, Gilad Yehudai, and Ohad Shamir. On the optimal memorization power of relu neural networks, 2021.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[16] Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: Interpreting attention in protein language models, 2021.

[17] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2023.

[18] Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L. Seltzer. Transformer-transducer: End-to-end speech recognition with self-attention, 2019.

[19] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity, 2019.

# A Appendix

## A.1 Transformer Architecture

Denote the input dimension as $d$, the output dimension as $d'$, the internal dimension as $m$, and the feed-forward dimension as $m'$ (we follow convention and use $m' = 4m$). The number of heads and layers in the transformer are denoted $h$ and $l$ respectively. The query, key, value, and weight matrices for the $i$th attention head and $j$th layer are denoted $Q_{i,j}, K_{i,j}, V_{i,j} \in \mathbb{R}^{d\times \frac{d}{h}}$, and $O_j \in \mathbb{R}^{d\times d}$ respectively.

For all experiments, we consider a non-causal transformer model. Namely, given the input sequence $\mathbf{x} \in \mathbb{R}^{n\times d}$, the model first applies the following transformations:

$$H_0 = \text{LayerNorm}(\phi(X)),$$

where $\phi(X) \in \mathbb{R}^{n\times m}$ denotes the tokenizer followed by layer normalization [2] with parameters $\gamma, \beta$:

$$\frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}(x)}}\gamma + \beta,$$

and dropout in the self-attention layers with a dropout probability of $p_1$, as well as in the feed-forward network with a dropout probability of $p_2$.[3]

The $j$th layer applies self-attention and a feed-forward transformation:

$$\begin{aligned} H_j' &= \text{LayerNorm}\left(\left(\left(\bigoplus_{i=1}^{h} \text{softmax}(H_{j-1}Q_{i,j}K_{i,j}^\top H_{j-1}^\top)H_{j-1}V_{i,j}\right)O_j\right)^\top\right) + H_{j-1}, \\ H_j &= \left(W_{1,j}\text{GeLU}(W_{2,j}(H_j')^\top)\right)^\top + H_{j-1}, \end{aligned}$$

where $\bigoplus_{i=1}^{h}$ denotes matrix concatenation, $W_{1,j}, W_{2,j} \in \mathbb{R}^{m'\times m}$ are the feed-forward weights, and $\text{GeLU}(\cdot)$ is the Gaussian Error Linear Unit [7]. Following the $L$-th layer, we apply a linear transformation to the output which will be the transformer's prediction:

$$f_{\text{trans}}(X) = H_L \cdot \text{Tanh}(W_{L+1}^\top),$$

where $W_{L+1} \in \mathbb{R}^{d'\times m}$ and $\text{Tanh}(\cdot)$ denotes the hyperbolic tangent activation function. Note that $d = d' = 1$ for the normal means problem.

# B Experimental Details

For all experiments, we use input sequences of length $n = 64$ with batch sizes of 4, early stopping, and no curriculum learning. During evaluation, models are tested on 512 samples drawn from both $\pi_S$ and $\pi_{\mathcal{N}}$. To stabilize training, models with a hidden dimension of size $m = 256$ are trained for at most 500,000 epochs with a learning rate of 0.0001 and weight decay of 0.0001. Models with $m = 4, 16, 64$ are trained for at most 100,000 epochs with a learning rate of 0.001.

[3] Across all experiments, we set $\gamma = 1$, $\beta = 1$, $p_1 = p_2 = 0$, as we did not observe a noticeable difference in the results with different values.

# C Variable Noise Setting

In addition to assessing the memorization capacity of the transformer on the set of training tasks $S$, we also vary the noise rate $\sigma^2$ and draw $\boldsymbol{\mu}$ from $\pi_{\mathcal{N}}$ to examine whether transformers learn predictors that perform well across different noise rates. Analogous to Section 2.1, the observables $\mathbf{x}$ are drawn from $\mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_n)$, where $\sigma^2$ is chosen randomly from a predefined set of noise rates $\sigma^2_{\text{noises}} = \{\sigma_1^2, \dots, \sigma_k^2\}$.[4]

In this setting, the risk of an estimator $f : \mathbb{R}^n \to \mathbb{R}^n$ for a fixed observation $(\mathbf{x}, \boldsymbol{\mu})$ is defined as:

$$\mathbb{E}_{\mathbf{x},\boldsymbol{\mu}}\big[\|f(\mathbf{x}) - \boldsymbol{\mu}\|_2^2 \mid \boldsymbol{\sigma} = \sigma\big].$$

Empirically, we consider the $\sigma^2$-normalized conditional empirical risk over $T$ samples:

$$\frac{1}{T\sigma^2} \sum_{i=1}^{T} \|f(\mathbf{x}_i) - \boldsymbol{\mu}_i\|_2^2.$$

During test time, the Bayes estimator that has access to the true noise rate (which is not necessarily in $\sigma_{\text{noises}}$) corresponds to the posterior mean function with known $\sigma^2$:

$$f_\sigma(\mathbf{x}) = \mathbb{E}[\boldsymbol{\mu} \mid \mathbf{x}, \boldsymbol{\sigma} = \sigma] = \frac{1}{\sigma^2 + 1}\mathbf{x}. \tag{5}$$

$f_\sigma$ is admissible, and we will refer to Eq. (5) as the *oracle estimator*.

Contrastingly, when the noise rate is unknown, a natural approach would be to make an informed guess of the noise rate:

$$f_{\text{B}}(\mathbf{x}) = \sum_{\sigma_i^2 \in \sigma^2_{\text{noises}}} \frac{\phi_{\boldsymbol{\mu},\sigma^2 I_n}(\mathbf{x})}{\sum_{\sigma'^2 \in \sigma^2_{\text{noises}}} \phi_{\boldsymbol{\mu}',\sigma'^2 I_n}(\mathbf{x})} \cdot \frac{1}{\sigma_i^2 + 1}\mathbf{x}.$$

Additionally, an empirical Bayes estimate of $\boldsymbol{\mu}$ using an unbiased estimate for the shrinkage factor $\frac{1}{\sigma^2+1}$, also known as the *James-Stein estimator*, can be employed:

$$f_{\text{JS}}(\mathbf{x}) = \left(1 - \frac{n-2}{\|\mathbf{x}\|_2^2}\right)\mathbf{x}.$$

It is noteworthy that the James-Stein estimator roughly corresponds to the shrinkage estimator in Section 2.1.

## C.1 Experiments

All models are trained on samples with noise rates $\sigma_{\text{noises}} = \{0.2, 0.8\}$ with squared Euclidean distance loss. Results are shown in Figure 4.

For smaller transformers, observe that the risk curves align with $f_{\text{JS}}$ across all noise rates (Figure 4a). As the James-Stein estimator is agnostic to the noise rate and always employs a fixed shrinkage, the alignment indicates the inability of smaller models to meaningfully adapt to variations in the noise rate. In larger models, we observe a notable transition from the models matching the James-Stein estimator to the Bayes estimator as training progresses (Figure 4b).

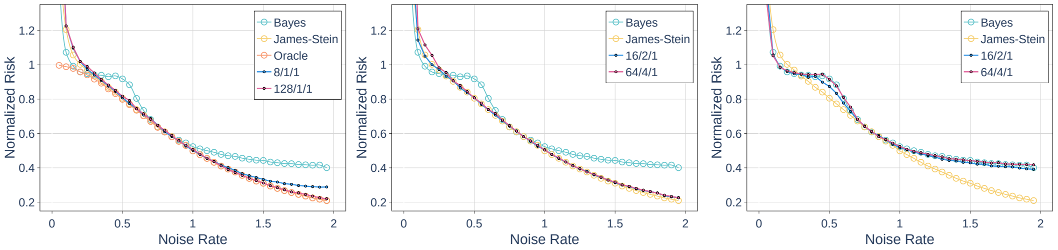
# D Supplementary Experiments

Here, we include results for additional experiments for the normal means problem.

## D.1 Varying Heads

In addition to varying the hidden dimension, number of layers, and training steps of the transformers, we consider varying the number of heads in the transformer (Figure 5). As changing the number of attention heads does not alter the number of parameters (Appendix A.1), it is perhaps unsurprising that the effect of the number of attention heads is minimal.

[4] For notational convenience, we use $\sigma_{\text{noises}}$ to refer to the standard deviations ($\sigma_{\text{noises}} = \{\sigma_1, \dots, \sigma_k\}$).



(a) First Result (Small models)

(b) Second Result (Large models)

Figure 4: Normalized empirical risk curves for various noise rates. For transformer models, $m/l/h$ respectively denote the hidden dimension size, the number of layers, and the number of heads. All transformer models are trained for 250,001 steps. In Figure 4b, the left plot shows risk curves after 137500 steps for $16/2/1$ and 24500 steps for $64/4/1$, while the right plot shows risk curves of last iteration. Additionally, the oracle estimator is omitted for clarity.

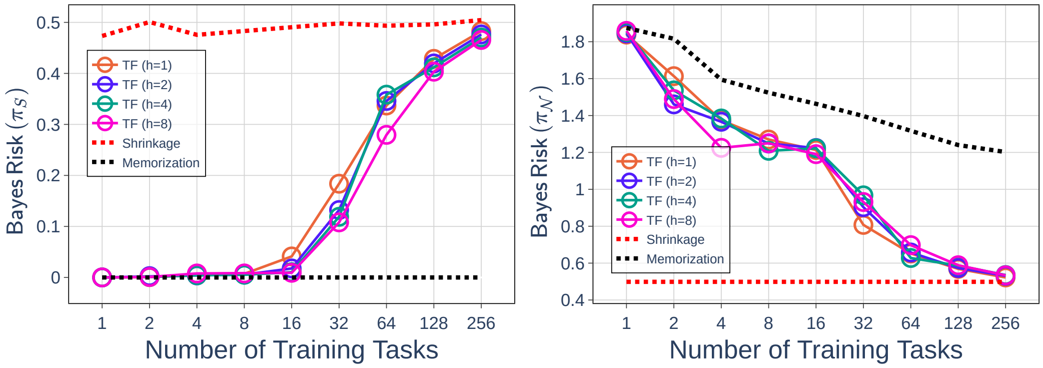


Figure 5: Relationship between the number of attention heads and the learned estimator. We use parameters $(m, l, t) = (16, 4, 100, 000)$ by default.

### D.2 Supplementary PCA Projections

Below, we include additional projections of the transformer's estimates (Figure 6 ~ 42). We use the same configuration as in Figure 2, with $(m, l, h) = (16, 2, 2)$ trained on $k = 8$ training tasks.

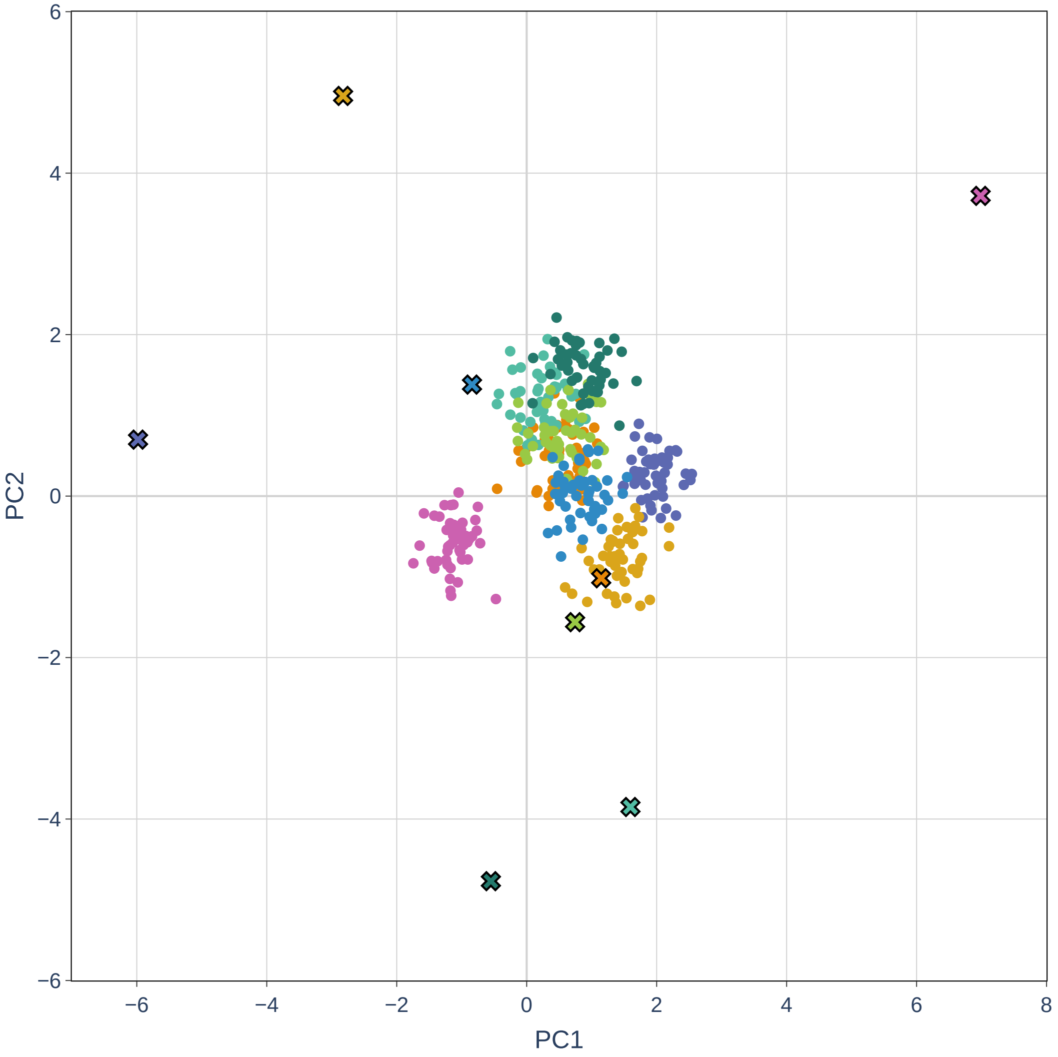


Figure 6: Epoch 0

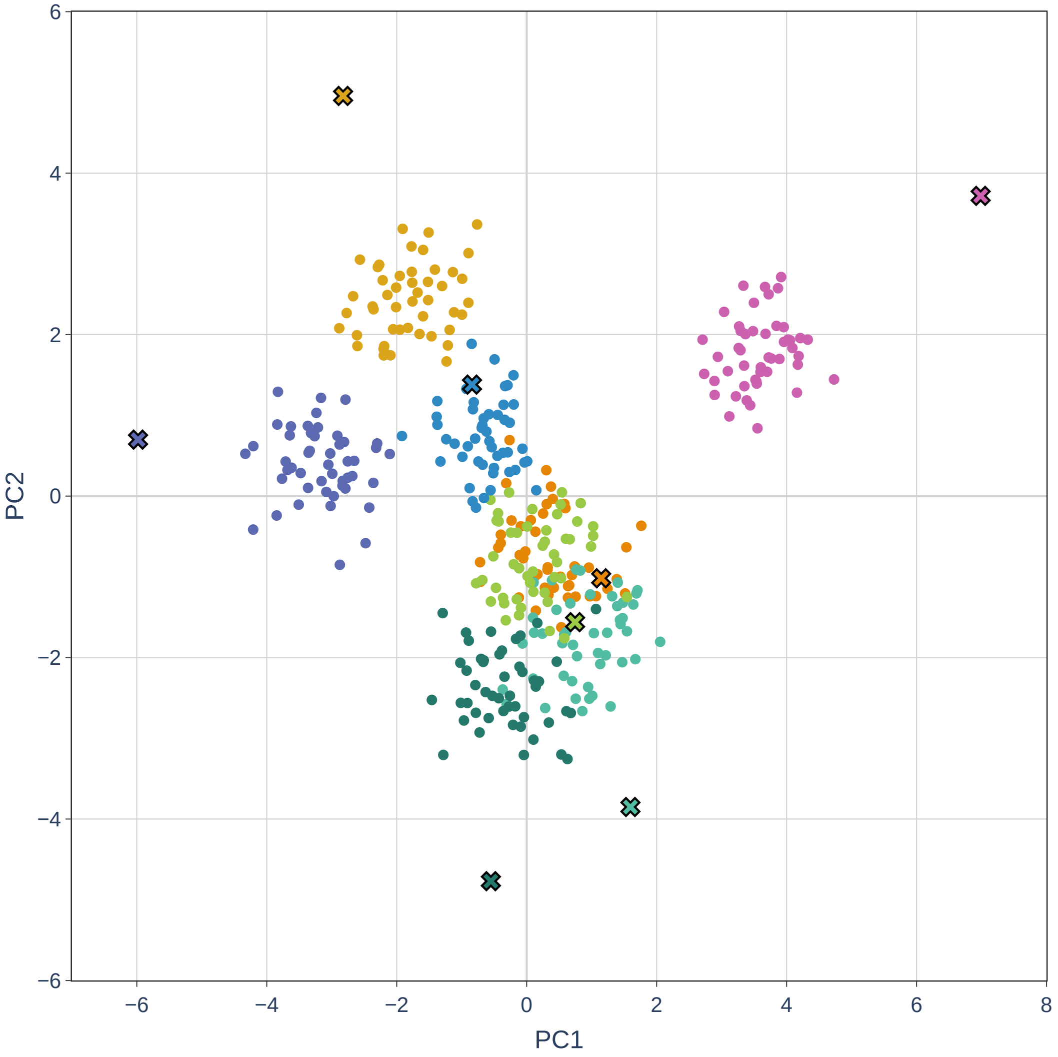


Figure 7: Epoch 1000

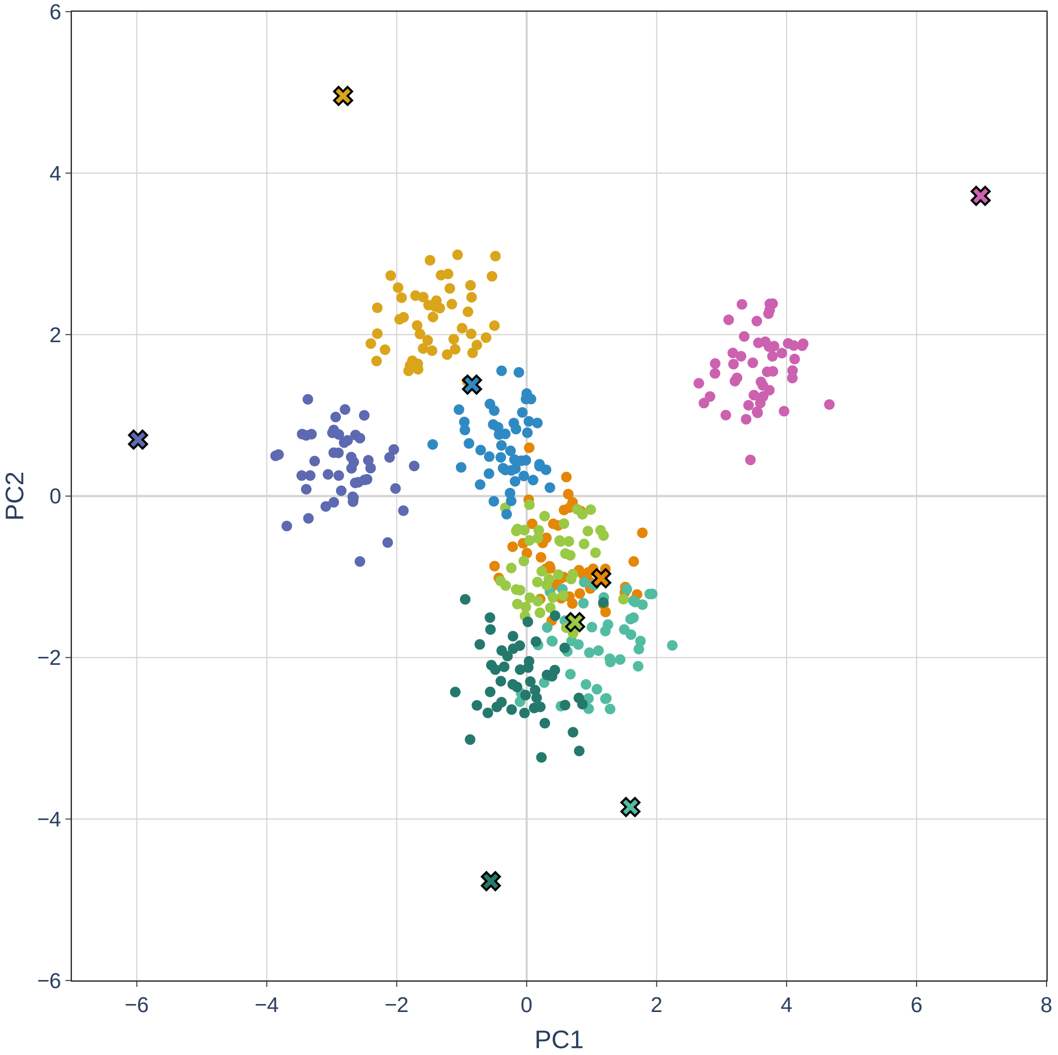


Figure 8: Epoch 2000

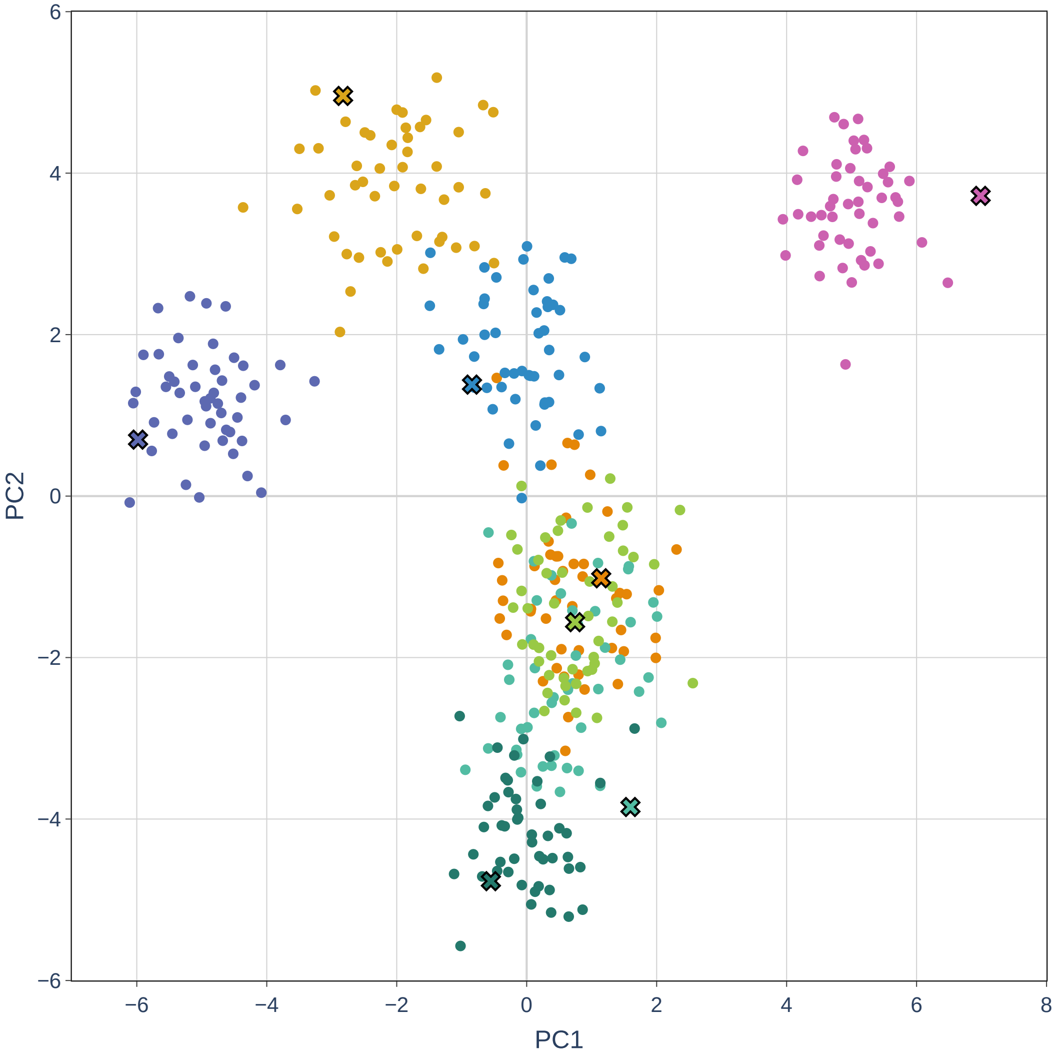


Figure 9: Epoch 3000

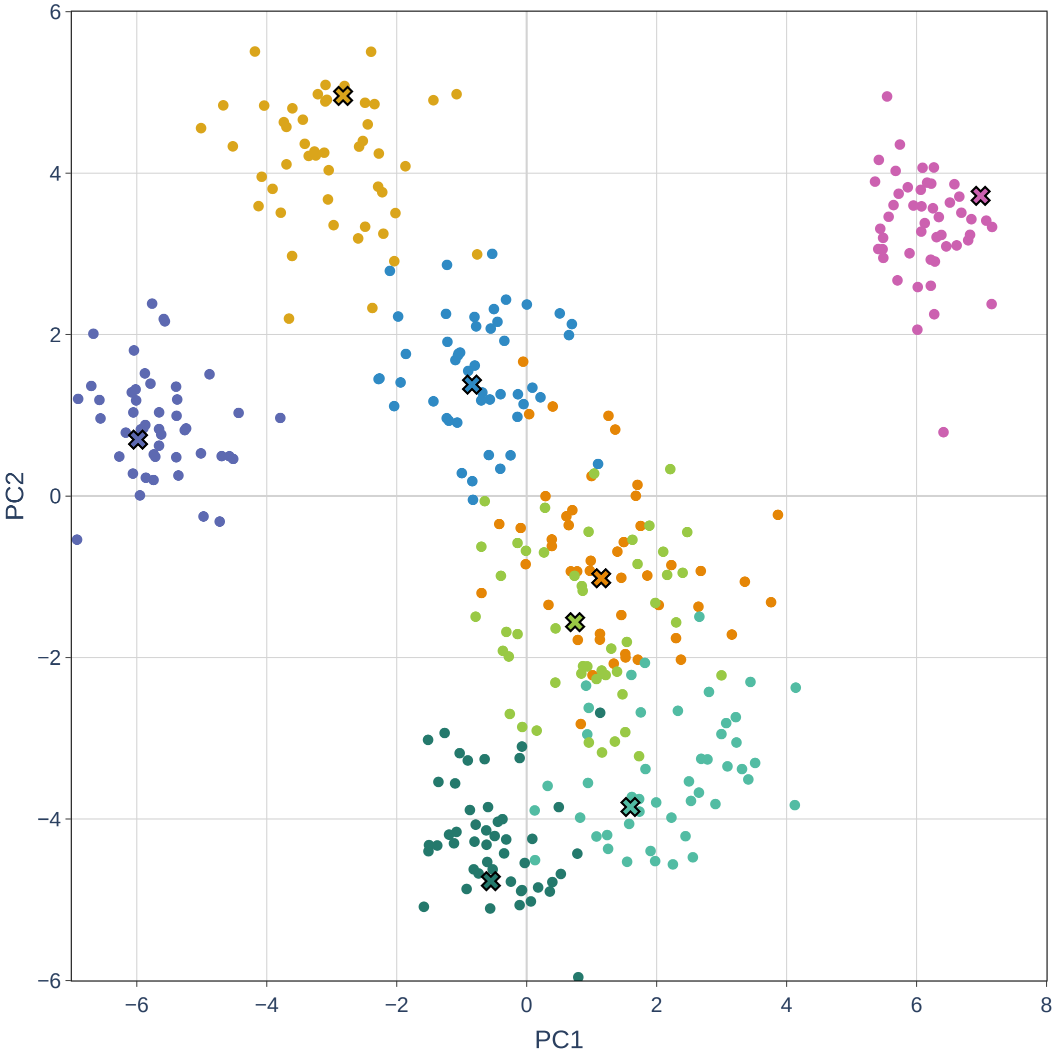


Figure 10: Epoch 4000

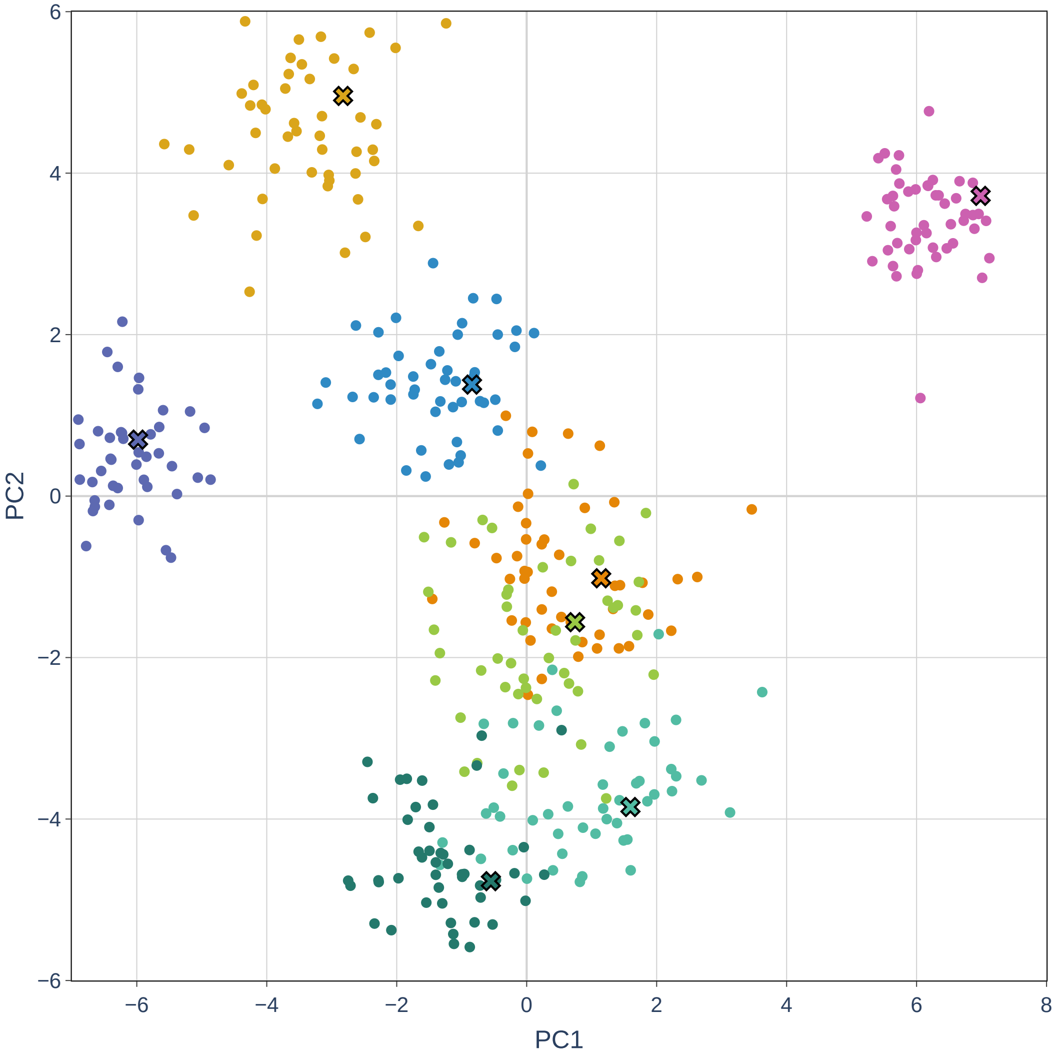


Figure 11: Epoch 5000

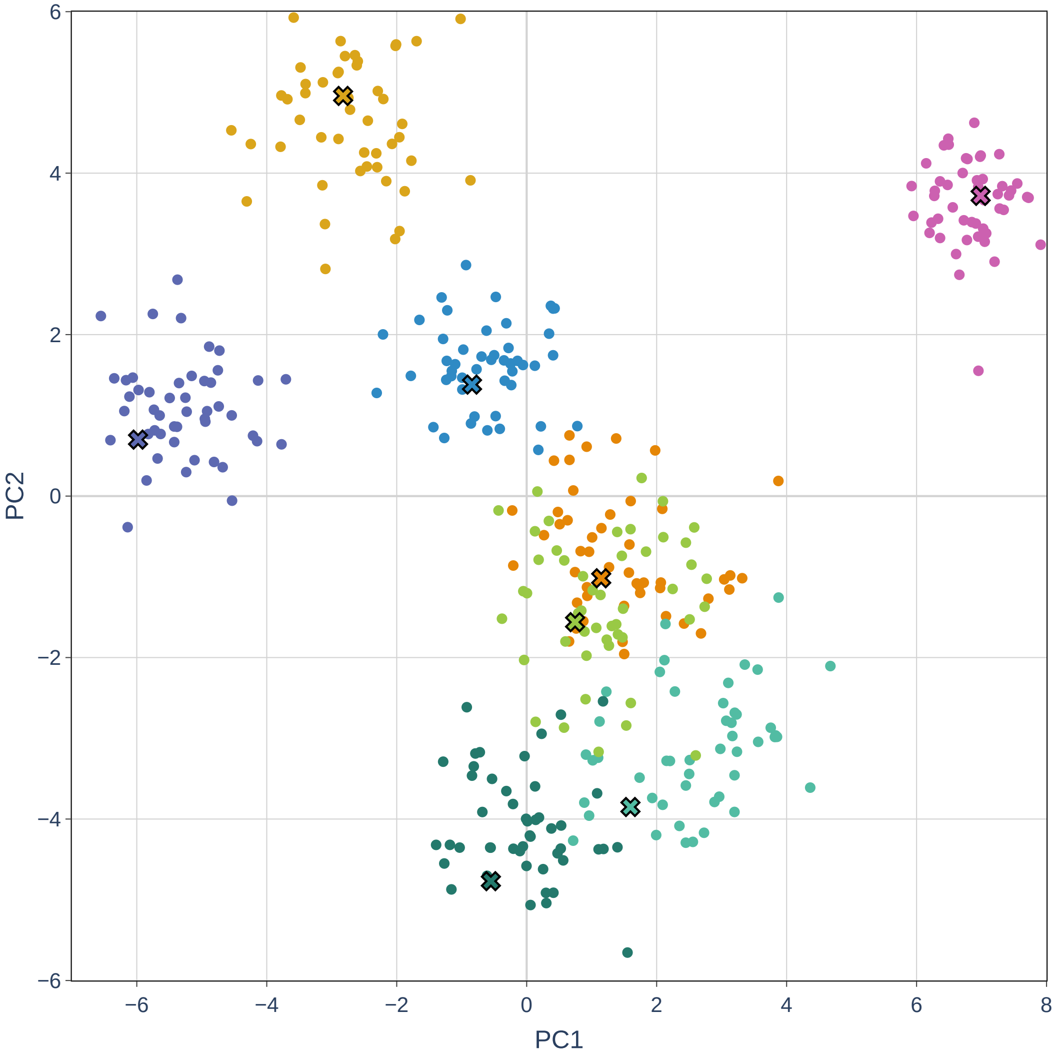


Figure 12: Epoch 6000

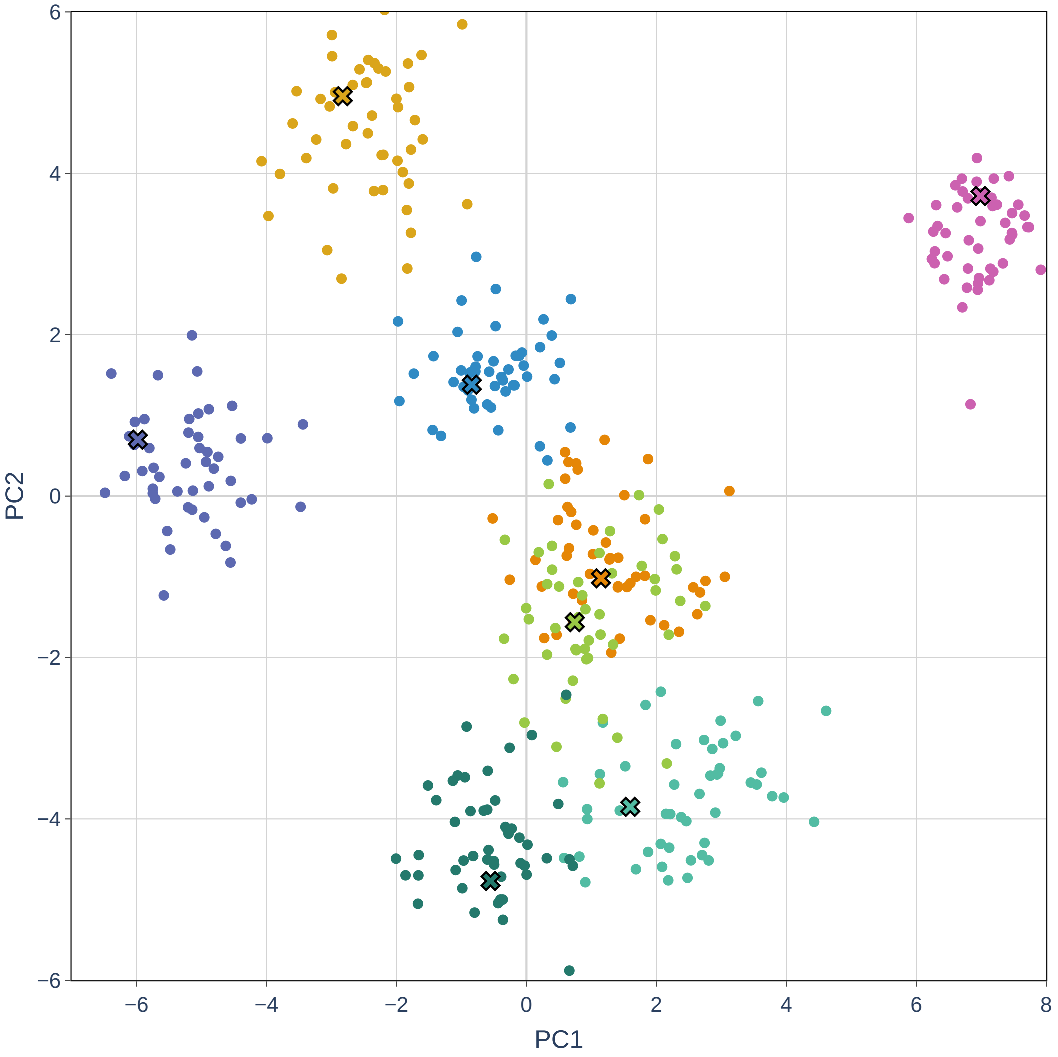


Figure 13: Epoch 7000

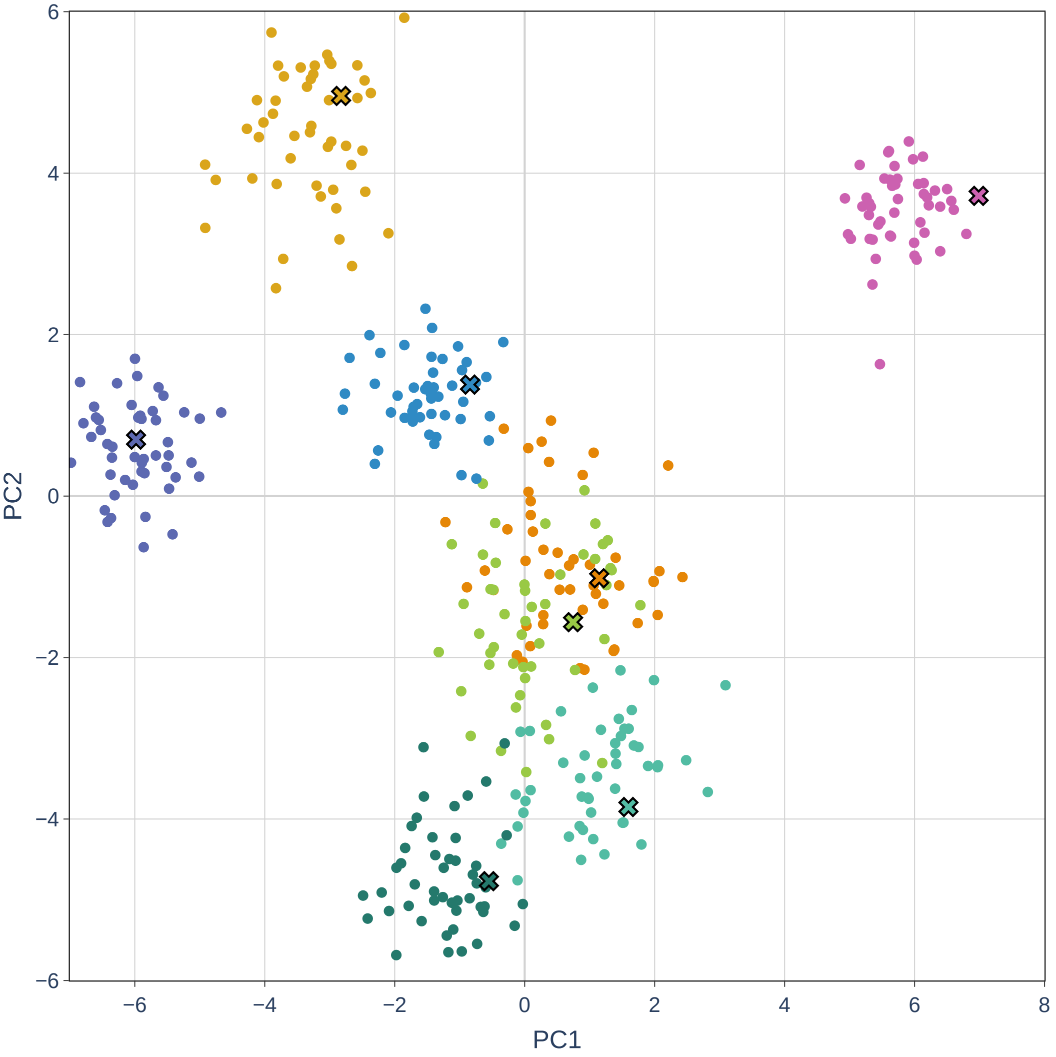


Figure 14: Epoch 8000

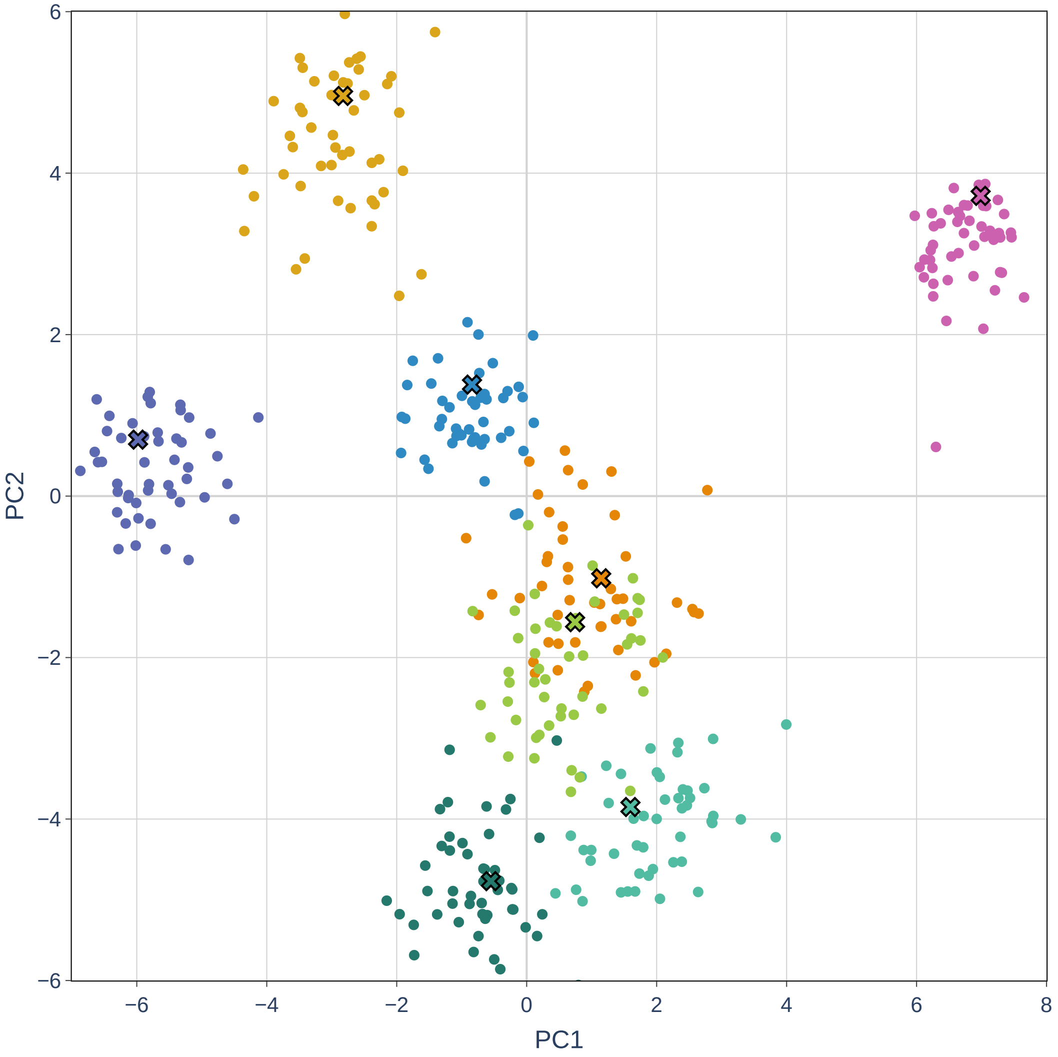


Figure 15: Epoch 9000

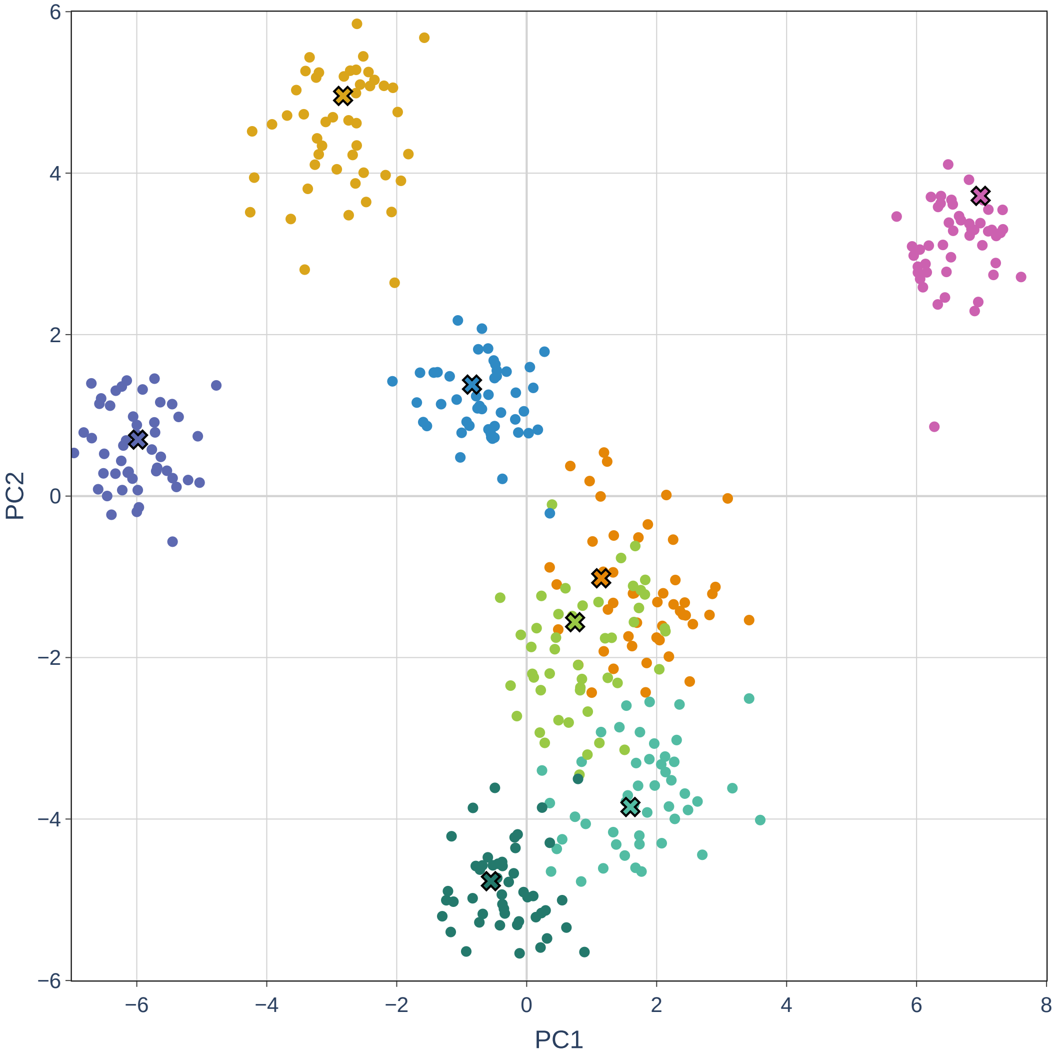


Figure 16: Epoch 10000

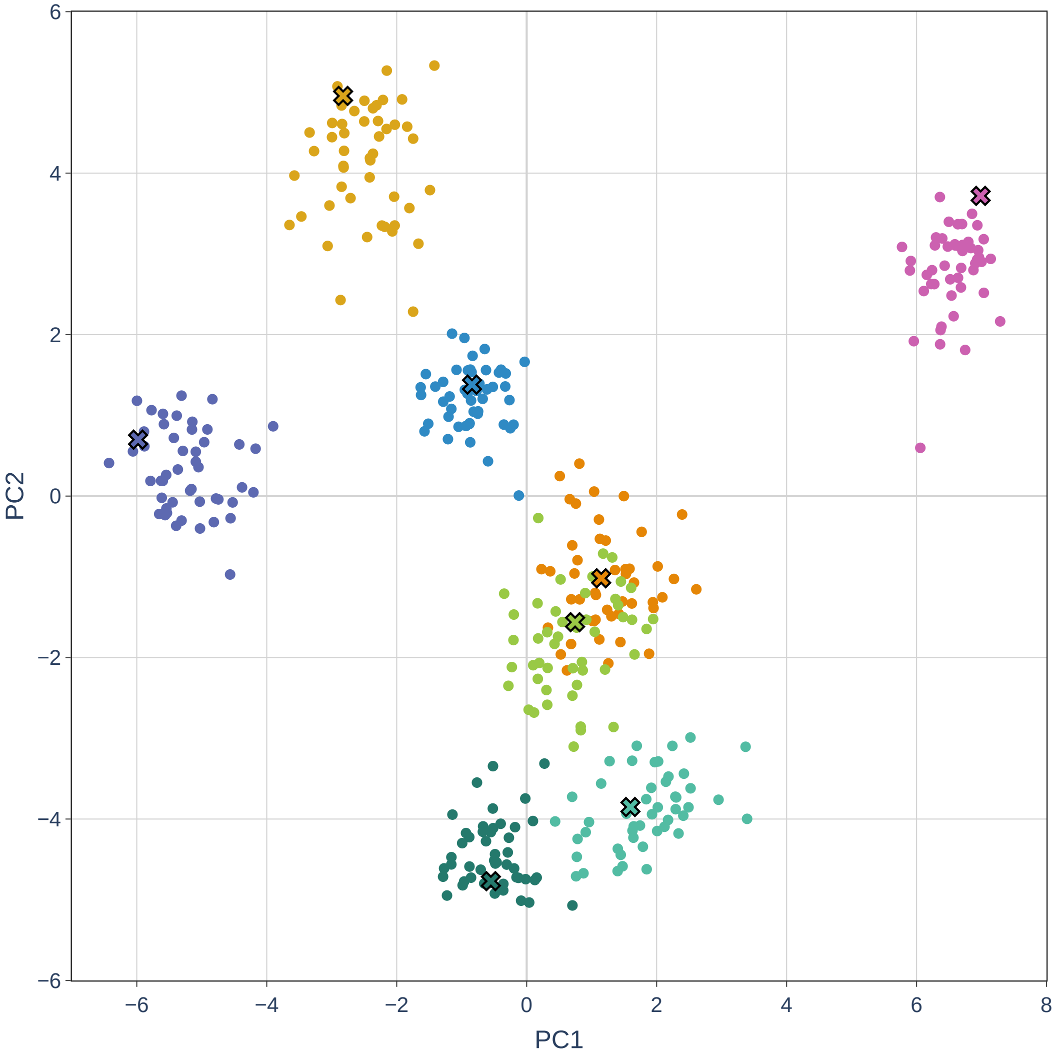


Figure 17: Epoch 11000

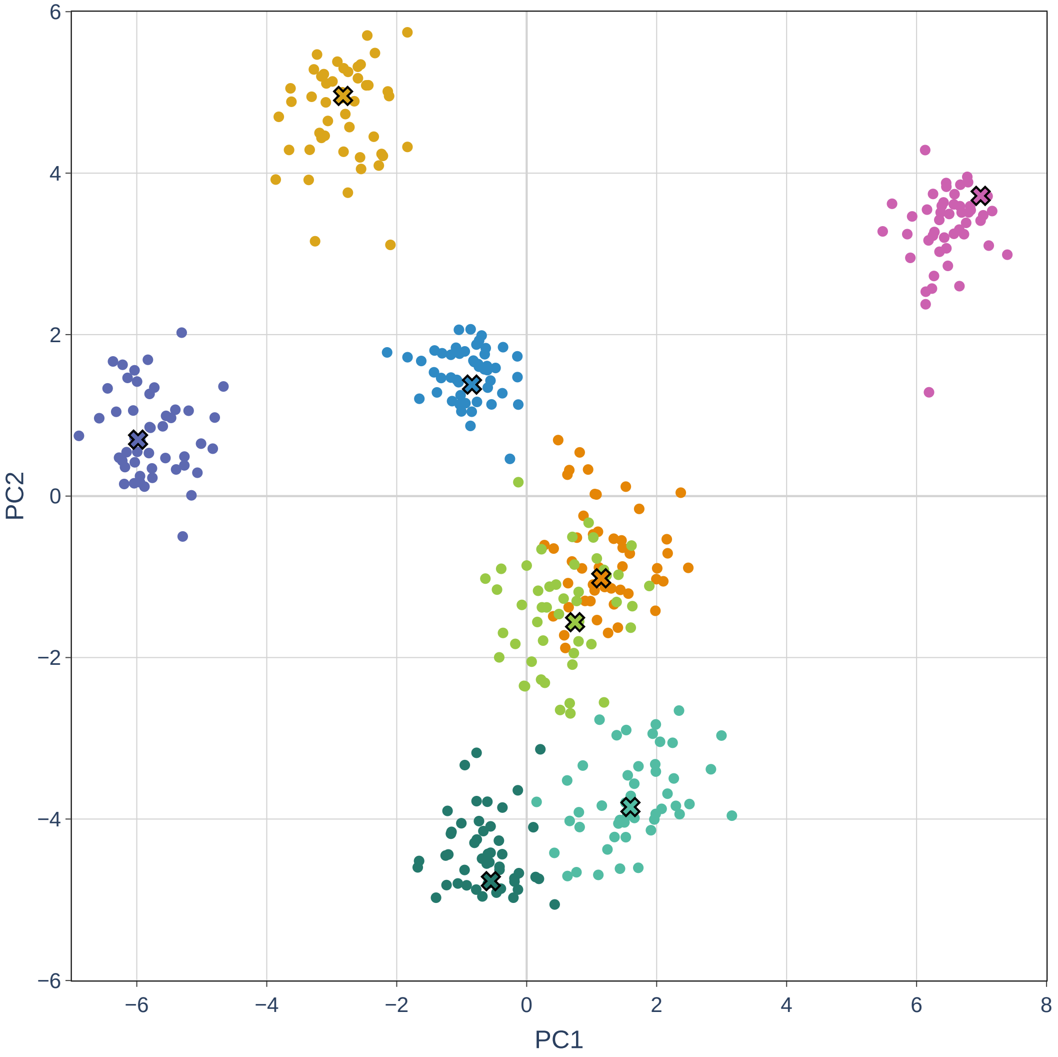


Figure 18: Epoch 12000

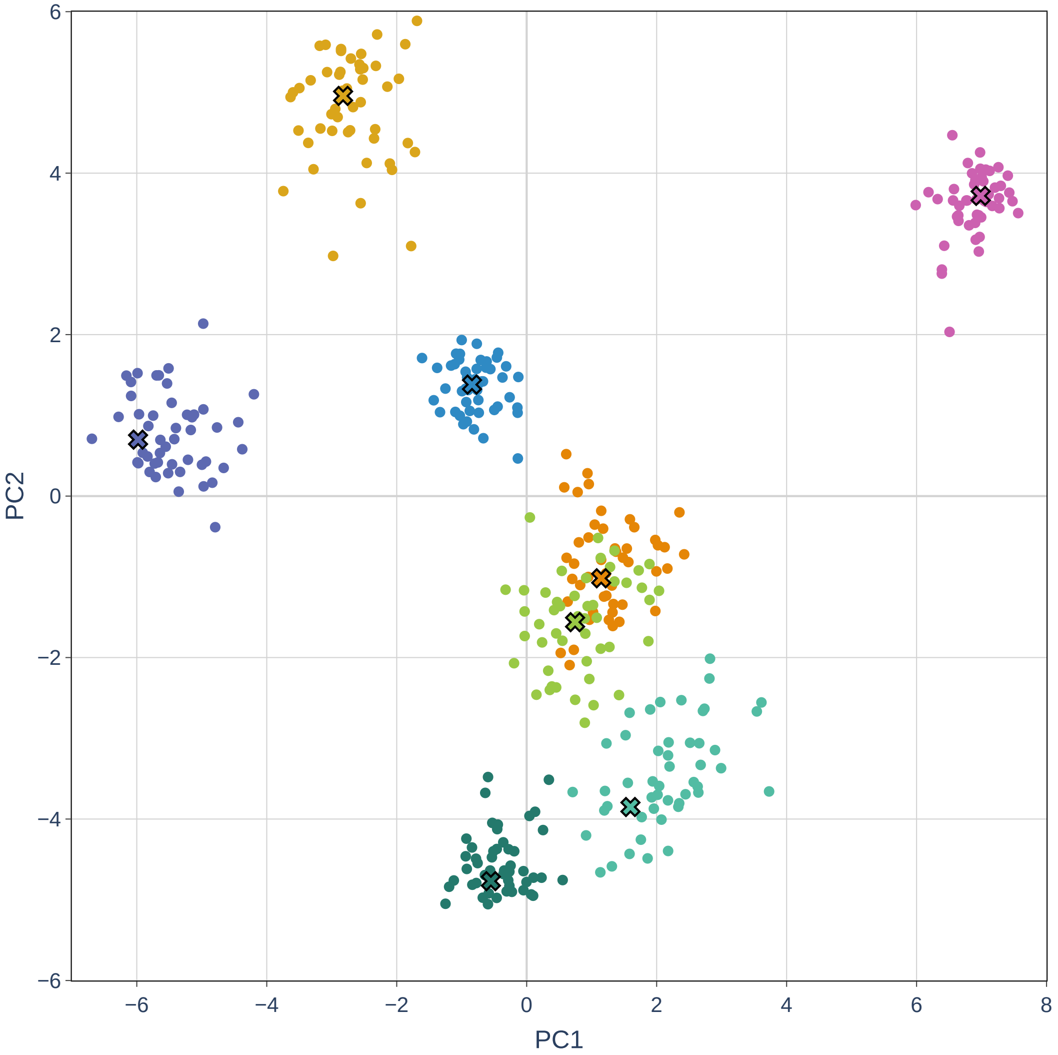


Figure 19: Epoch 13000

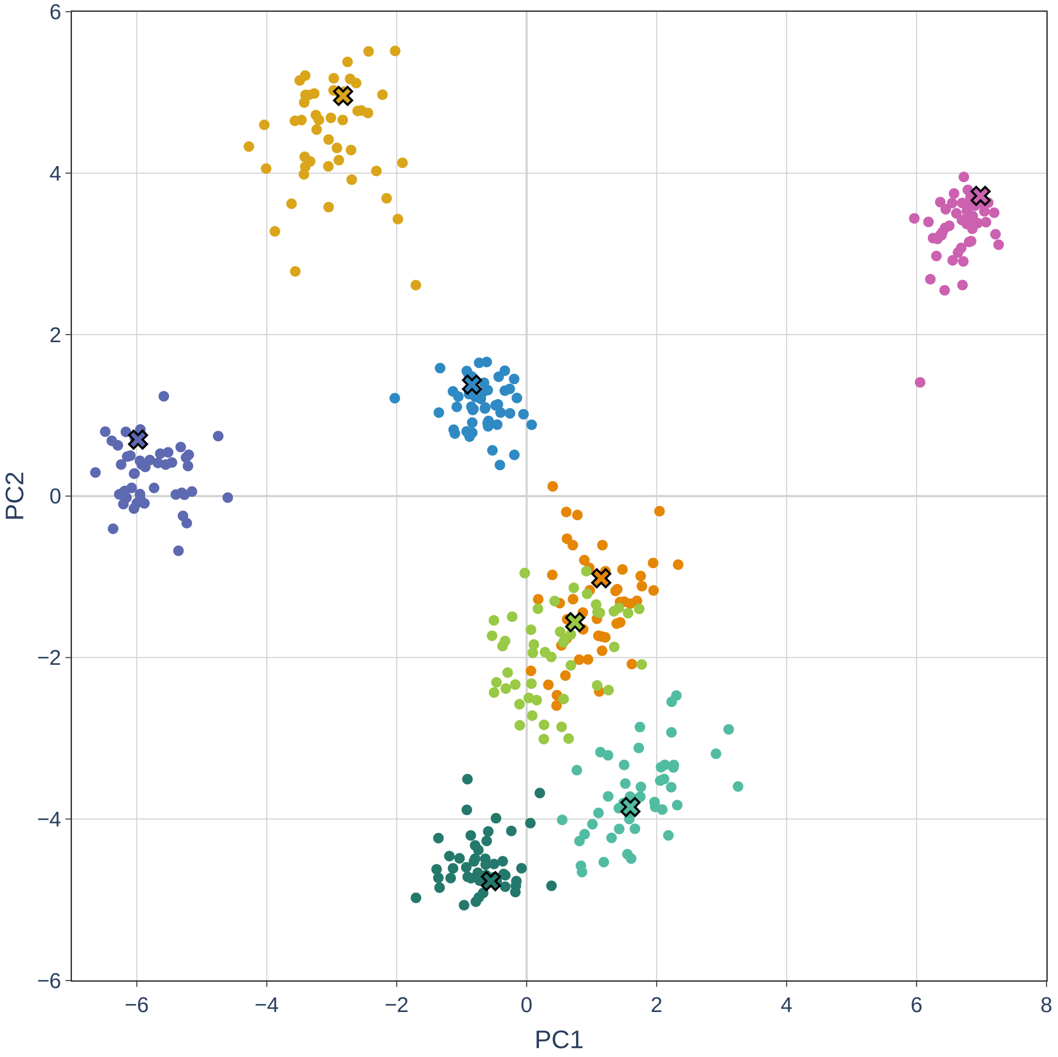


Figure 20: Epoch 14000

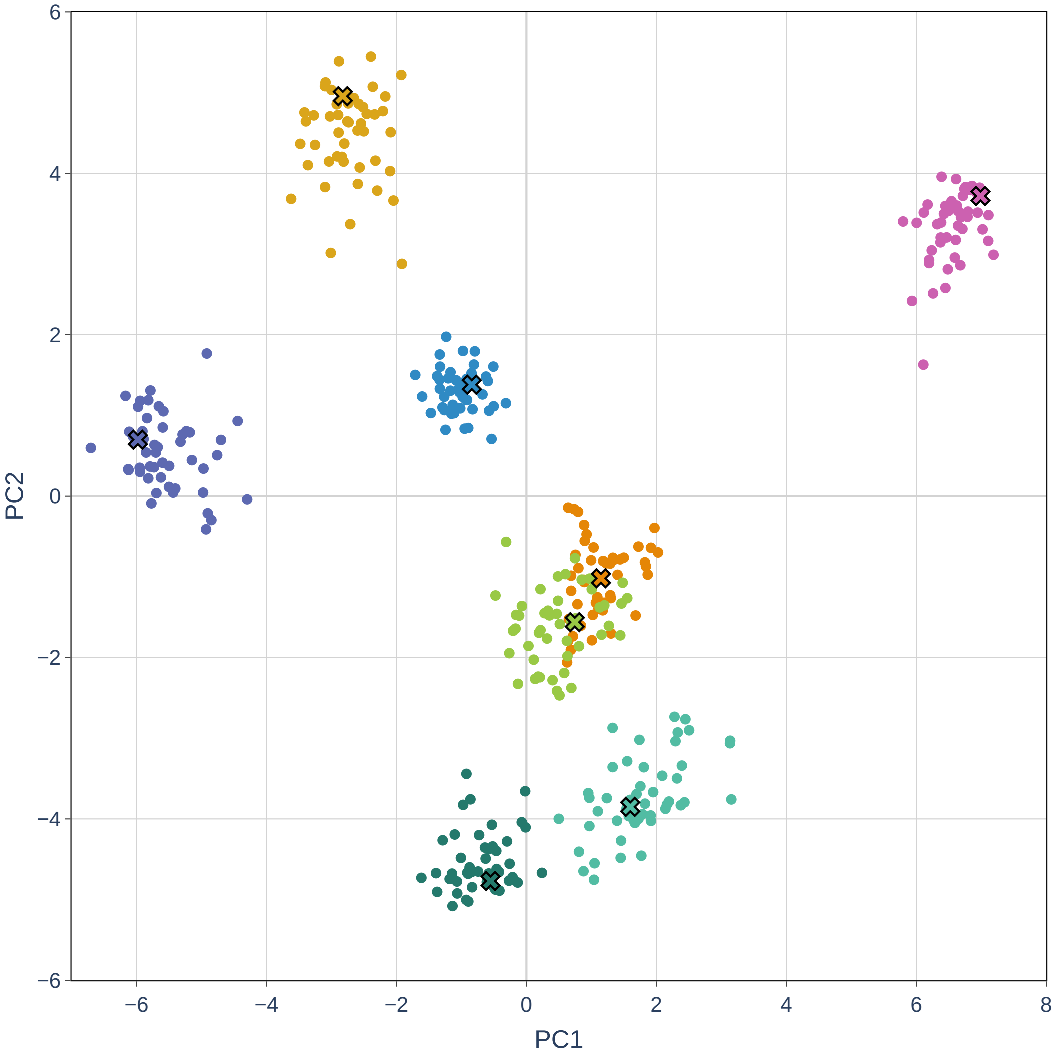


Figure 21: Epoch 15000

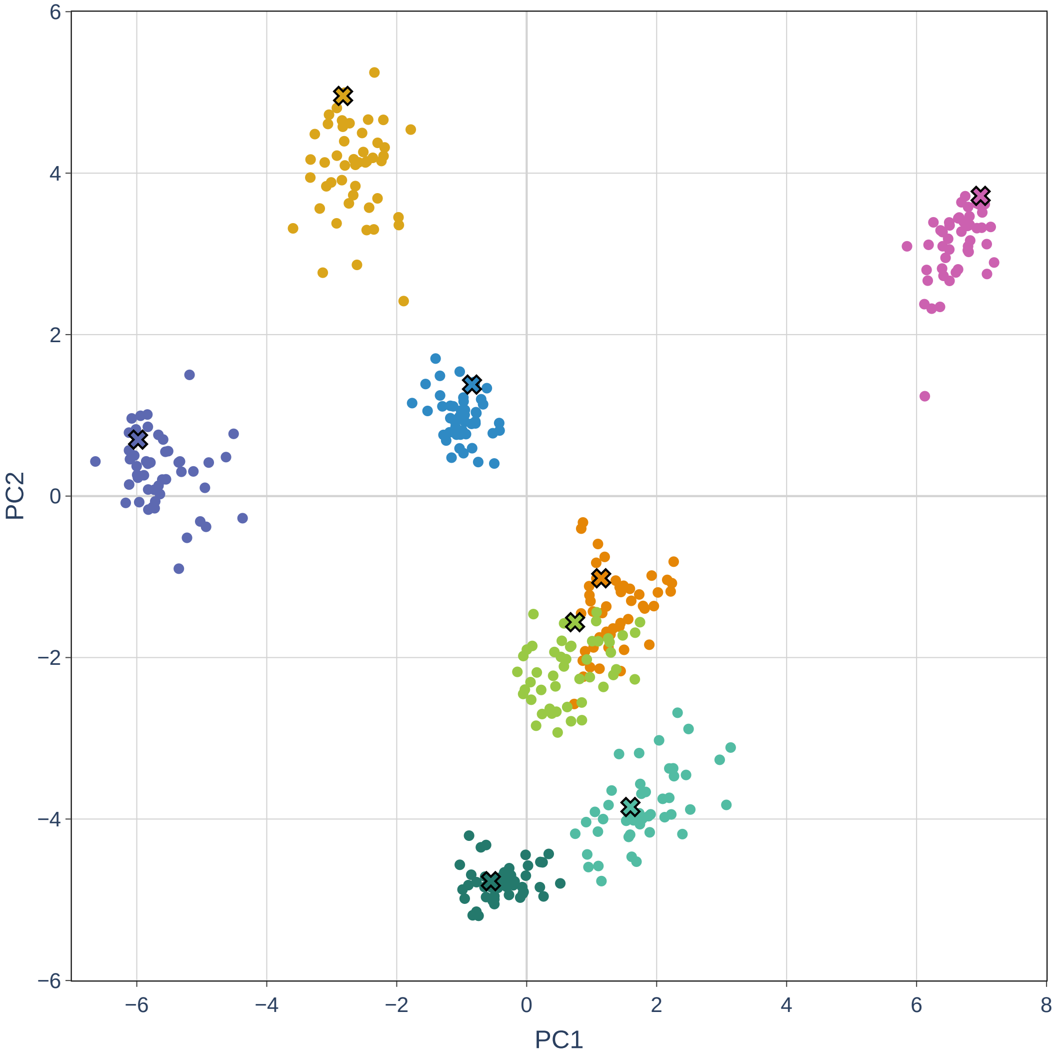


Figure 22: Epoch 16000

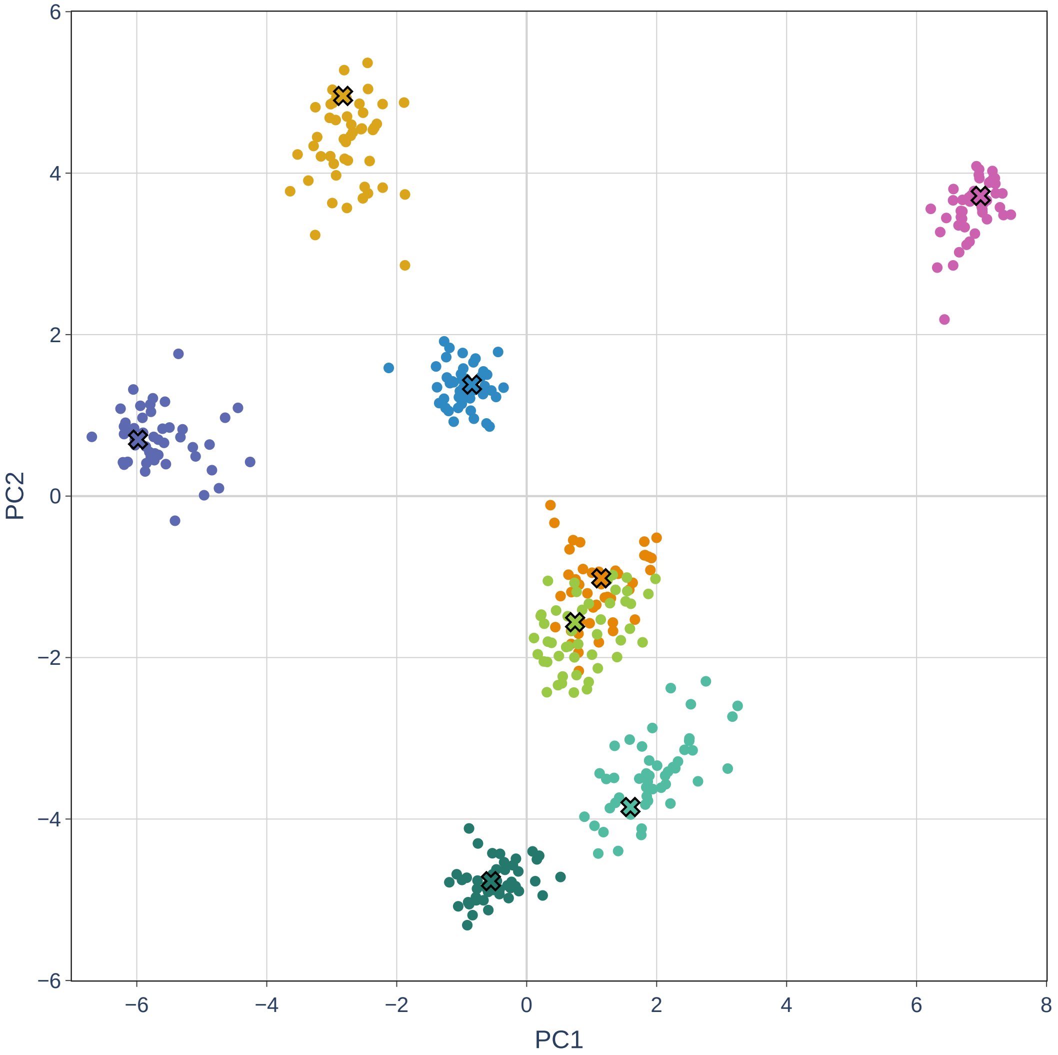


Figure 23: Epoch 17000

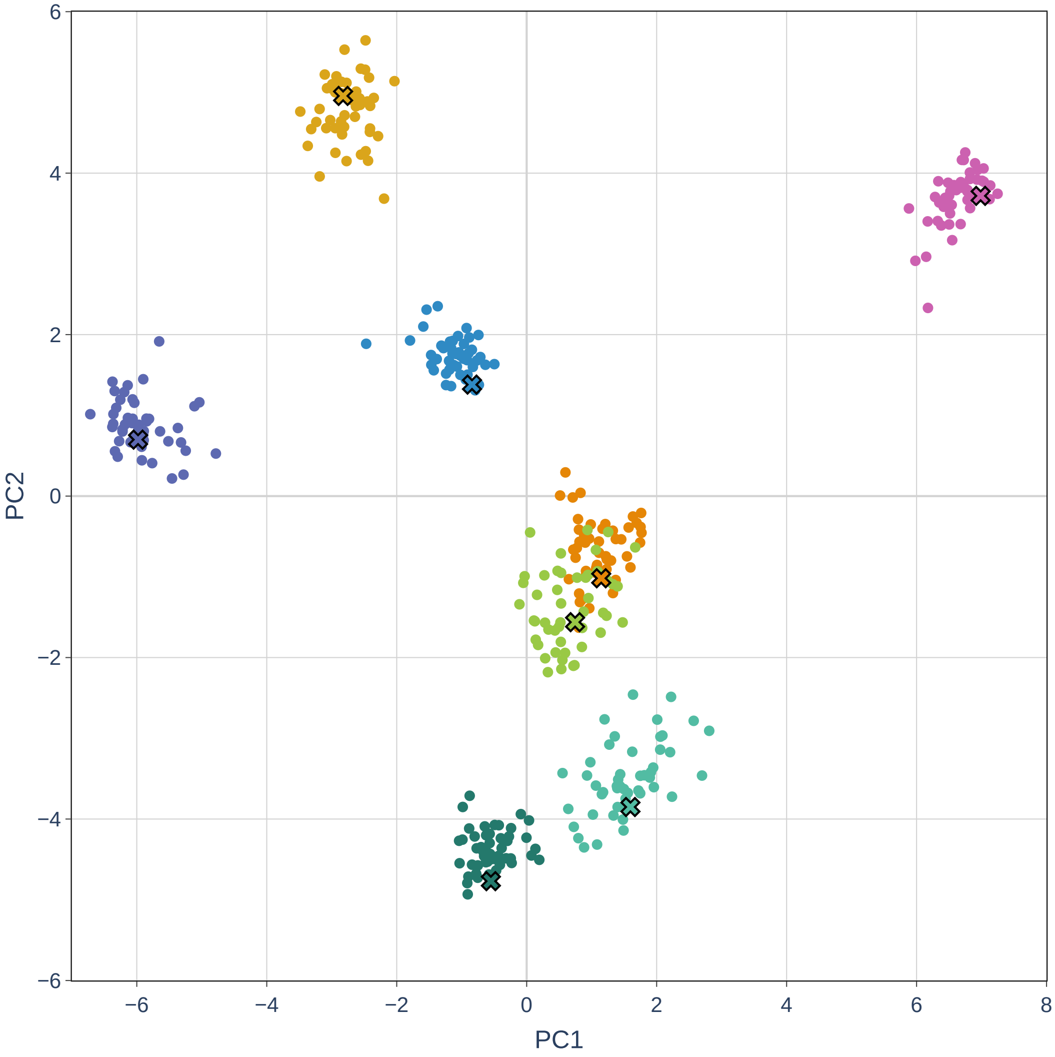


Figure 24: Epoch 18000

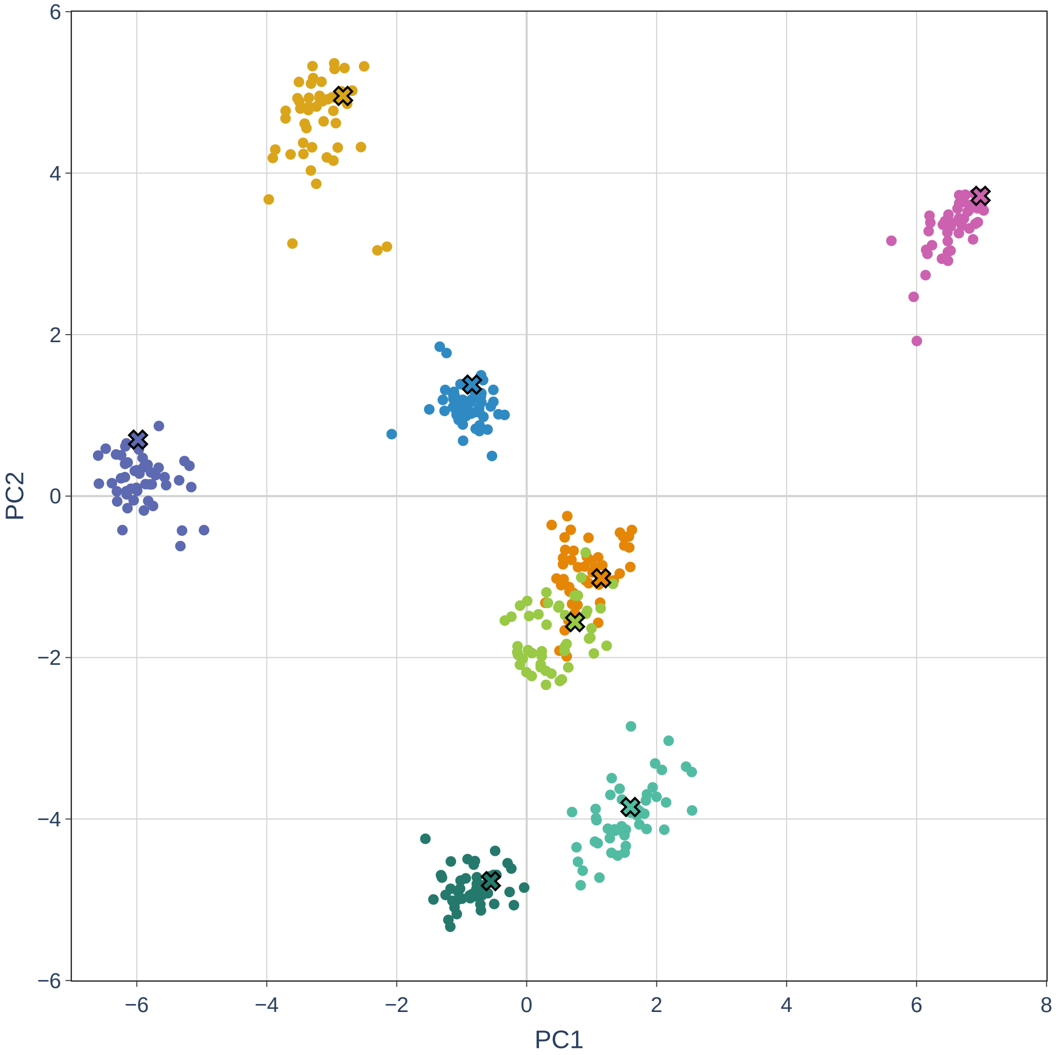


Figure 25: Epoch 19000

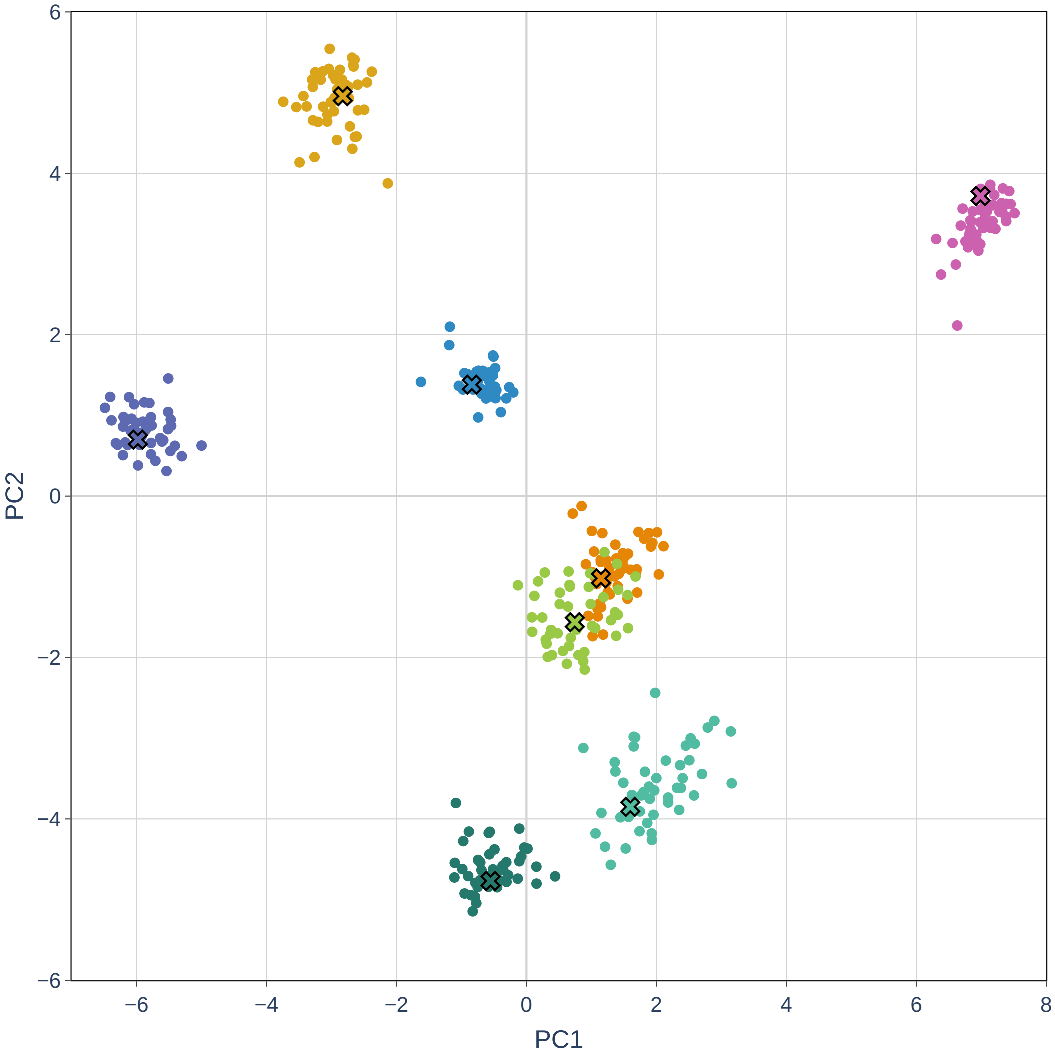


Figure 26: Epoch 20000

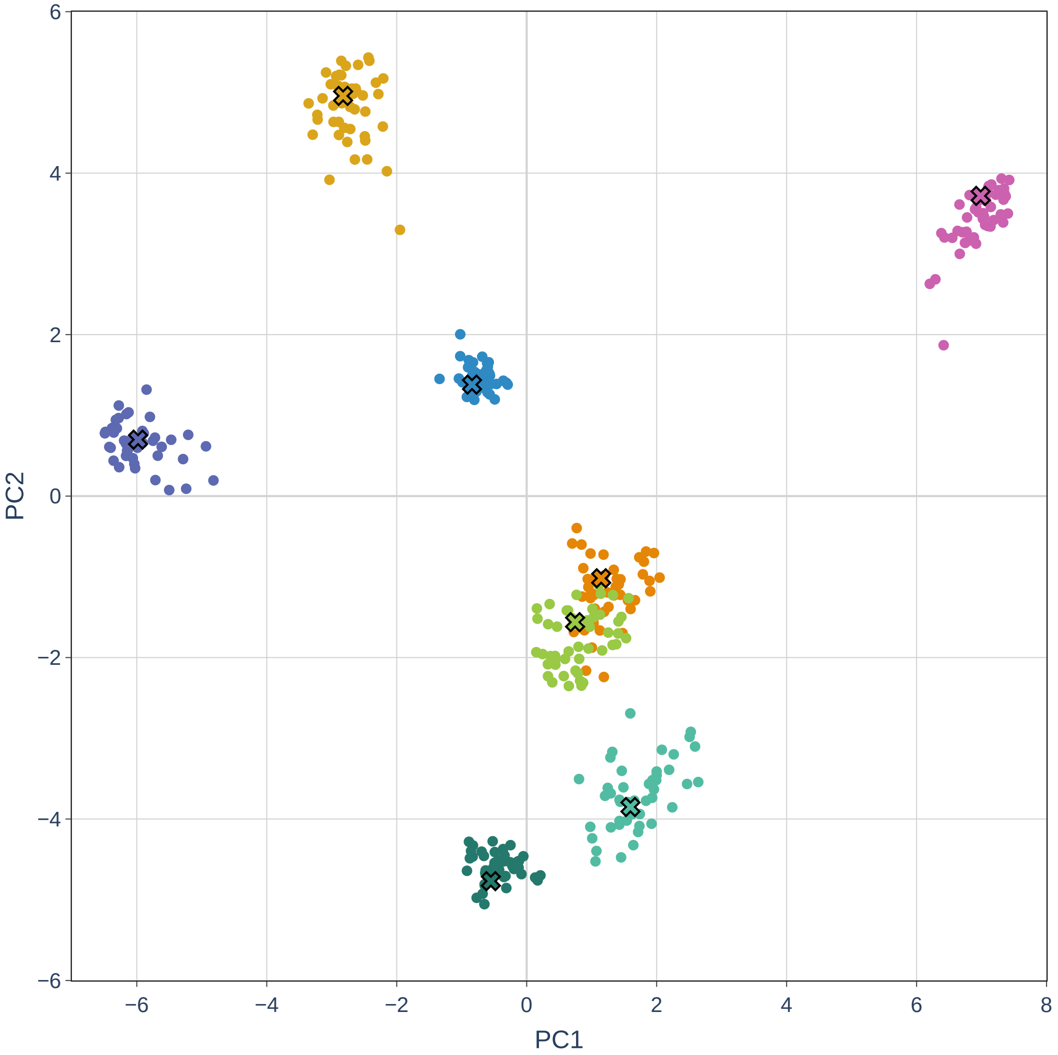


Figure 27: Epoch 21000

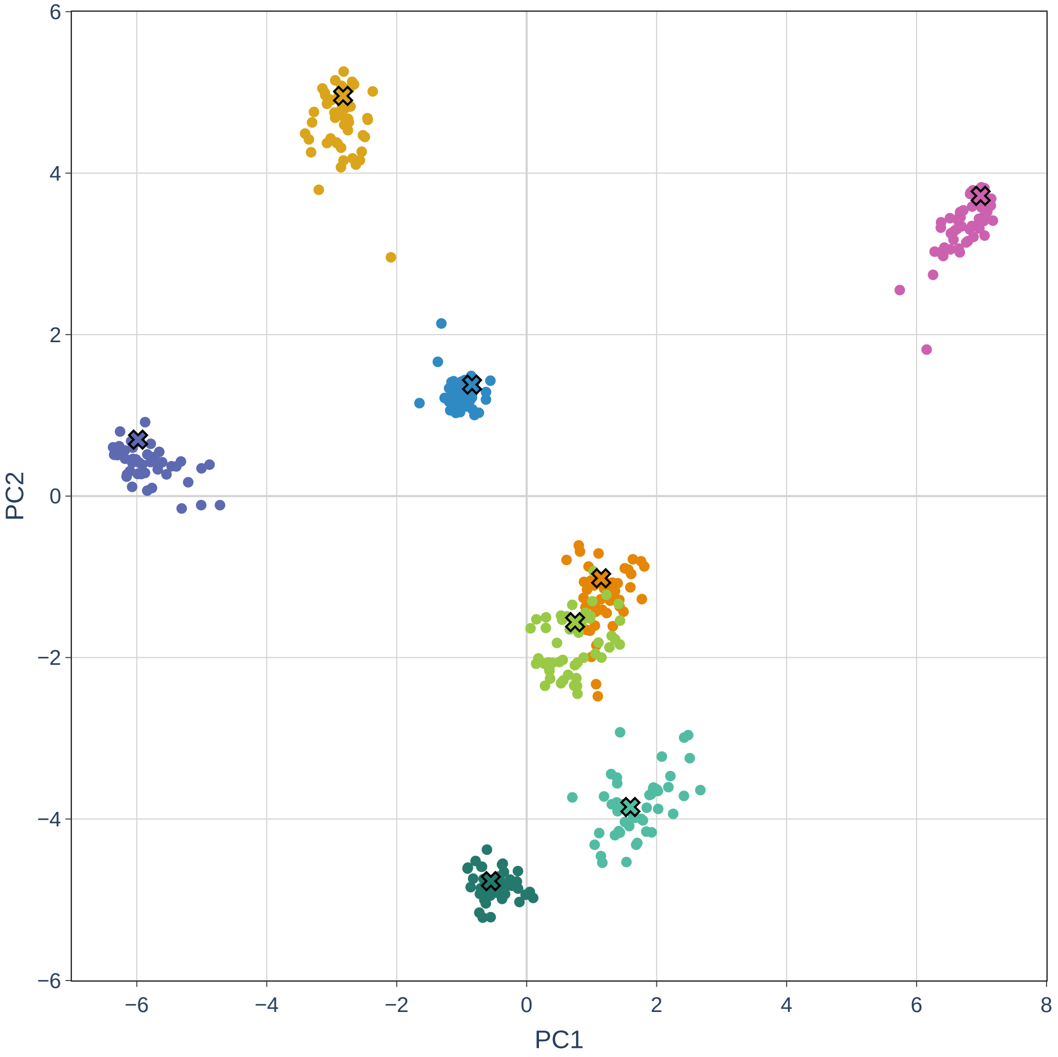


Figure 28: Epoch 22000

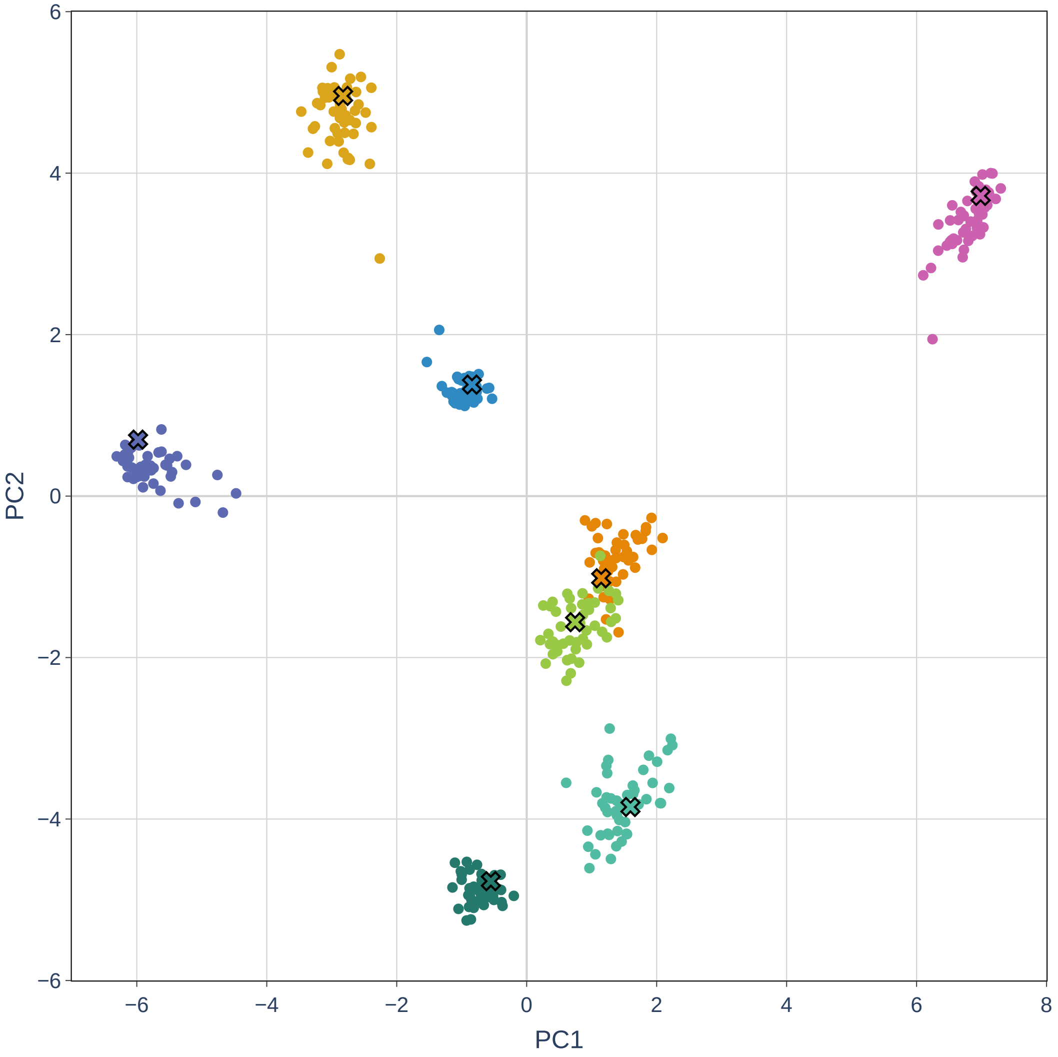


Figure 29: Epoch 23000

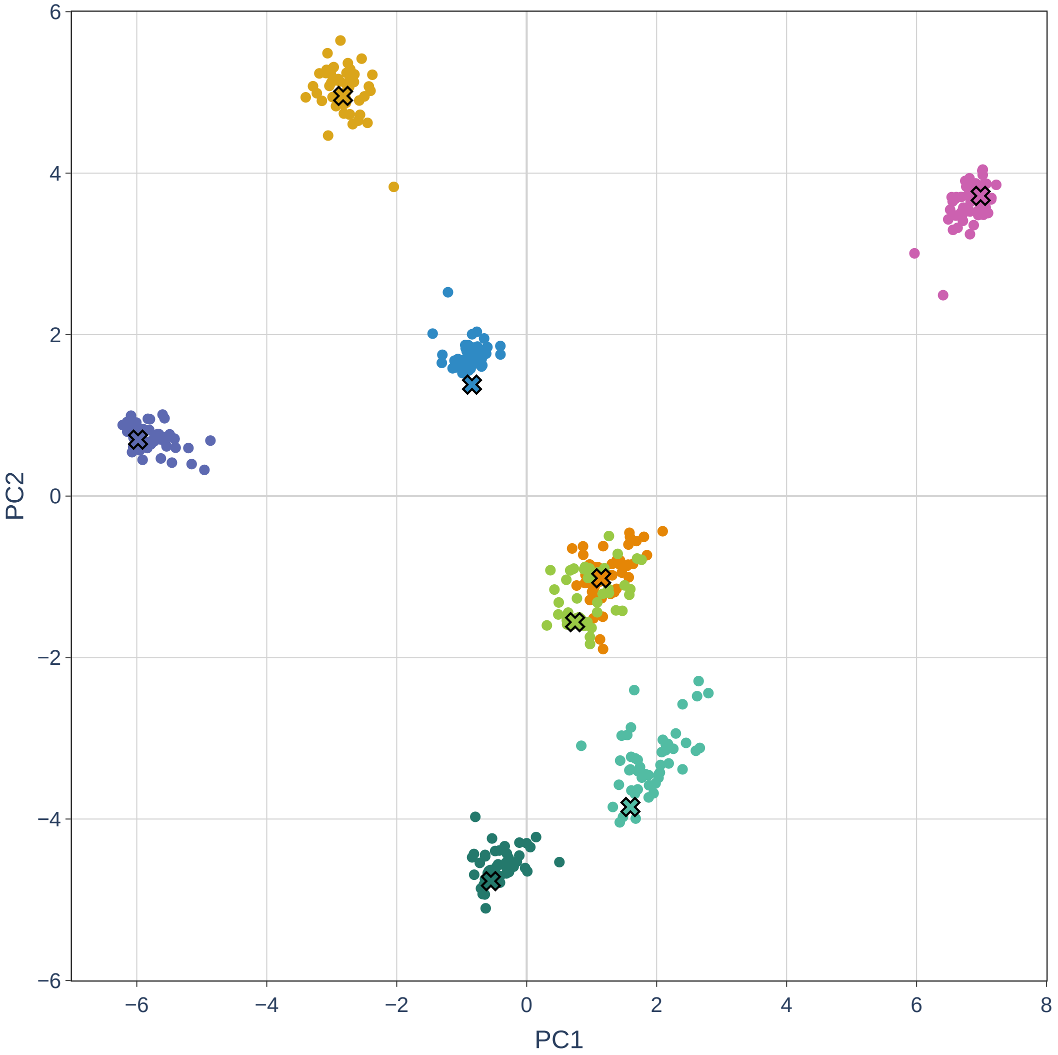


Figure 30: Epoch 24000

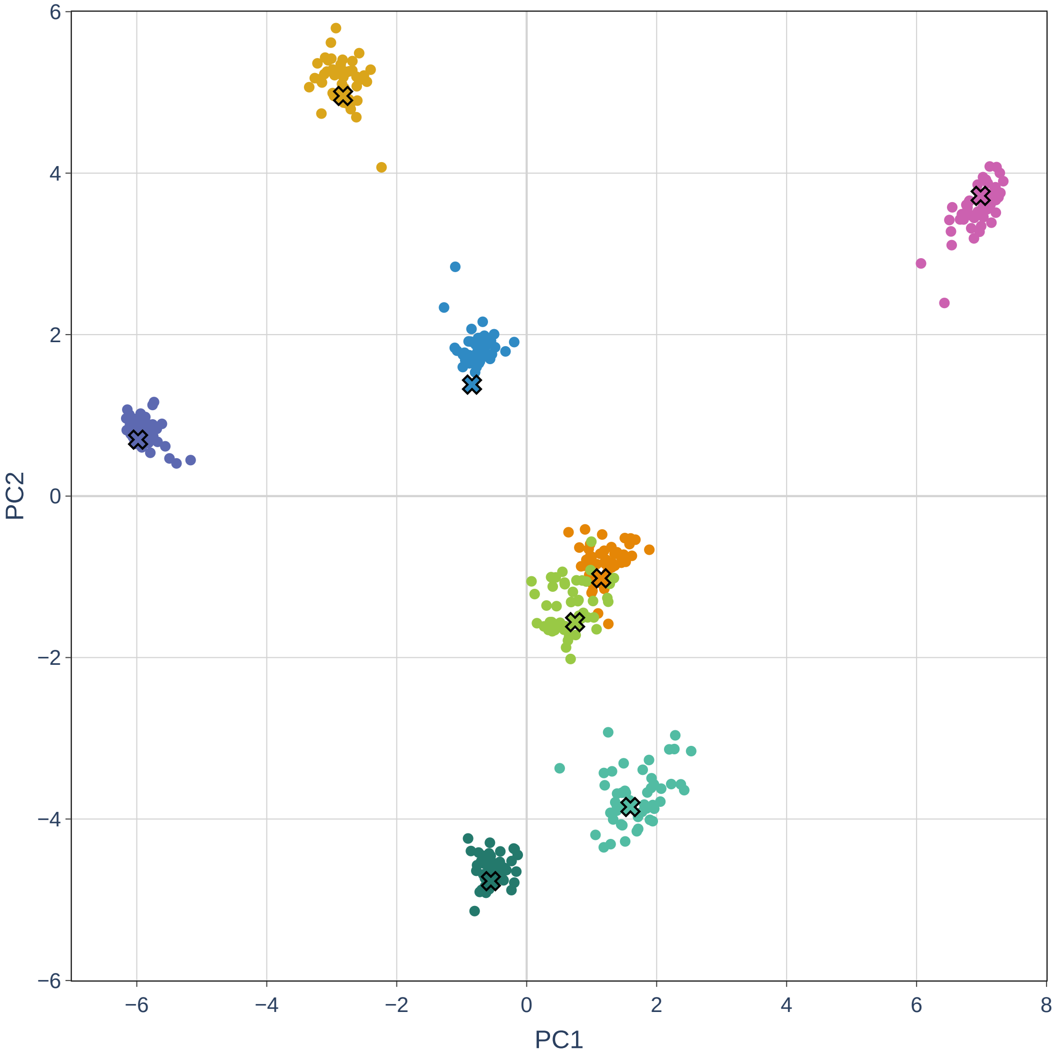


Figure 31: Epoch 25000

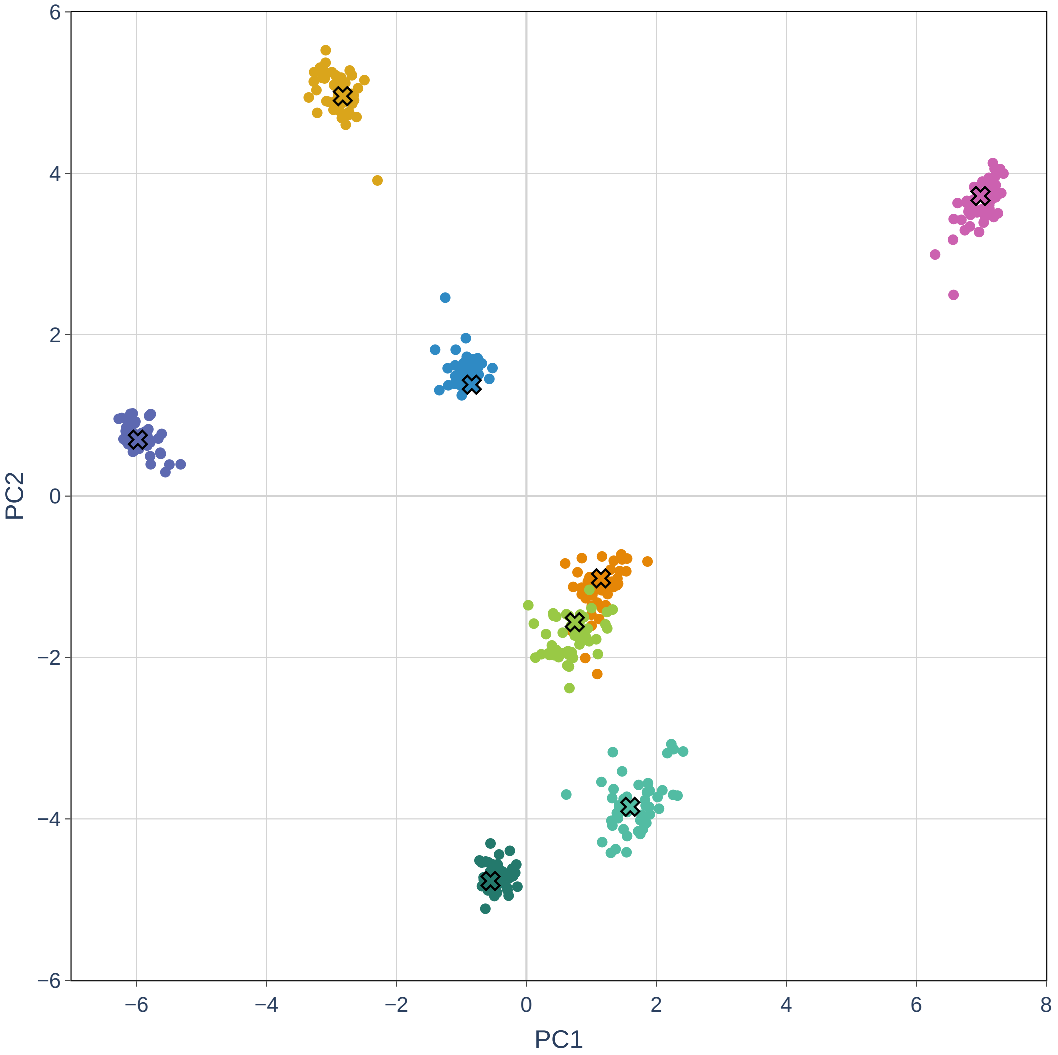


Figure 32: Epoch 26000

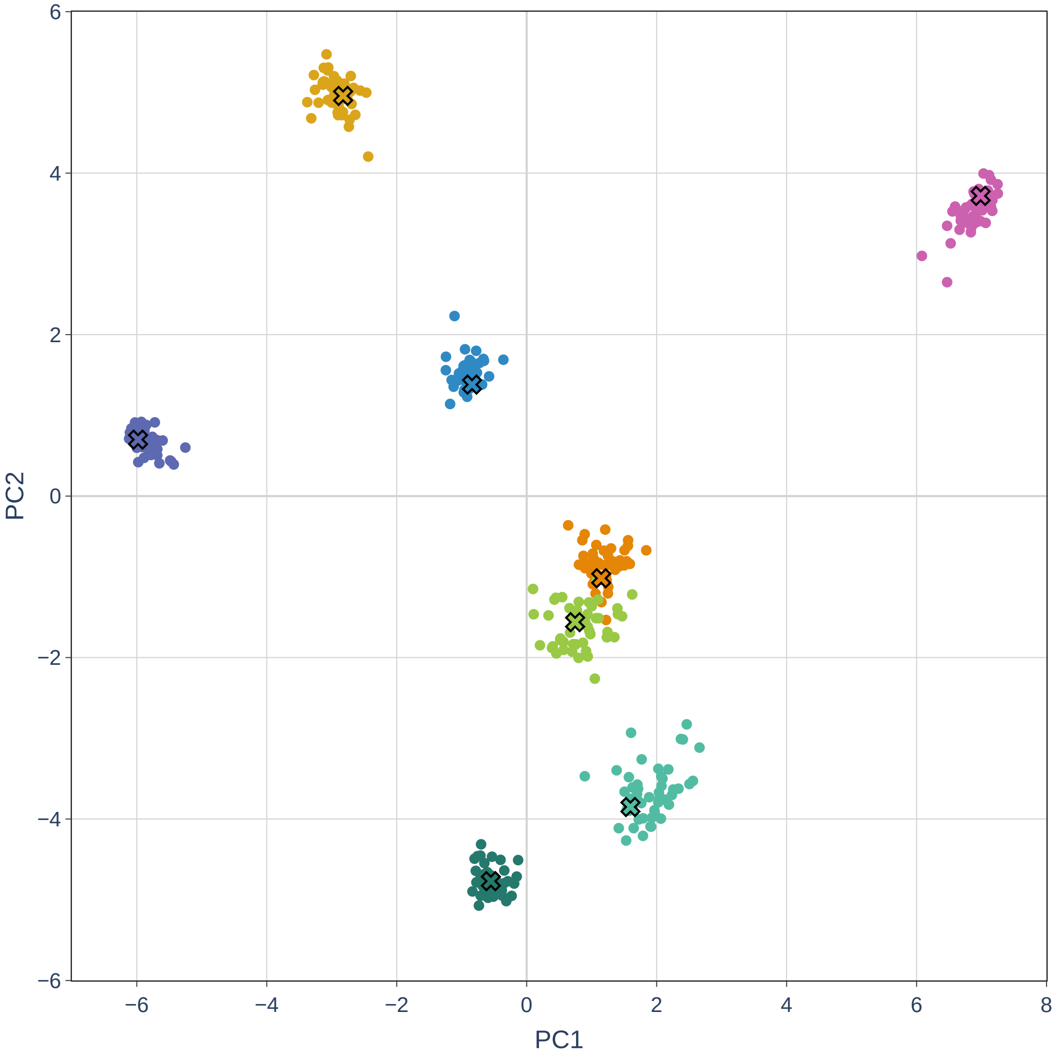


Figure 33: Epoch 27000

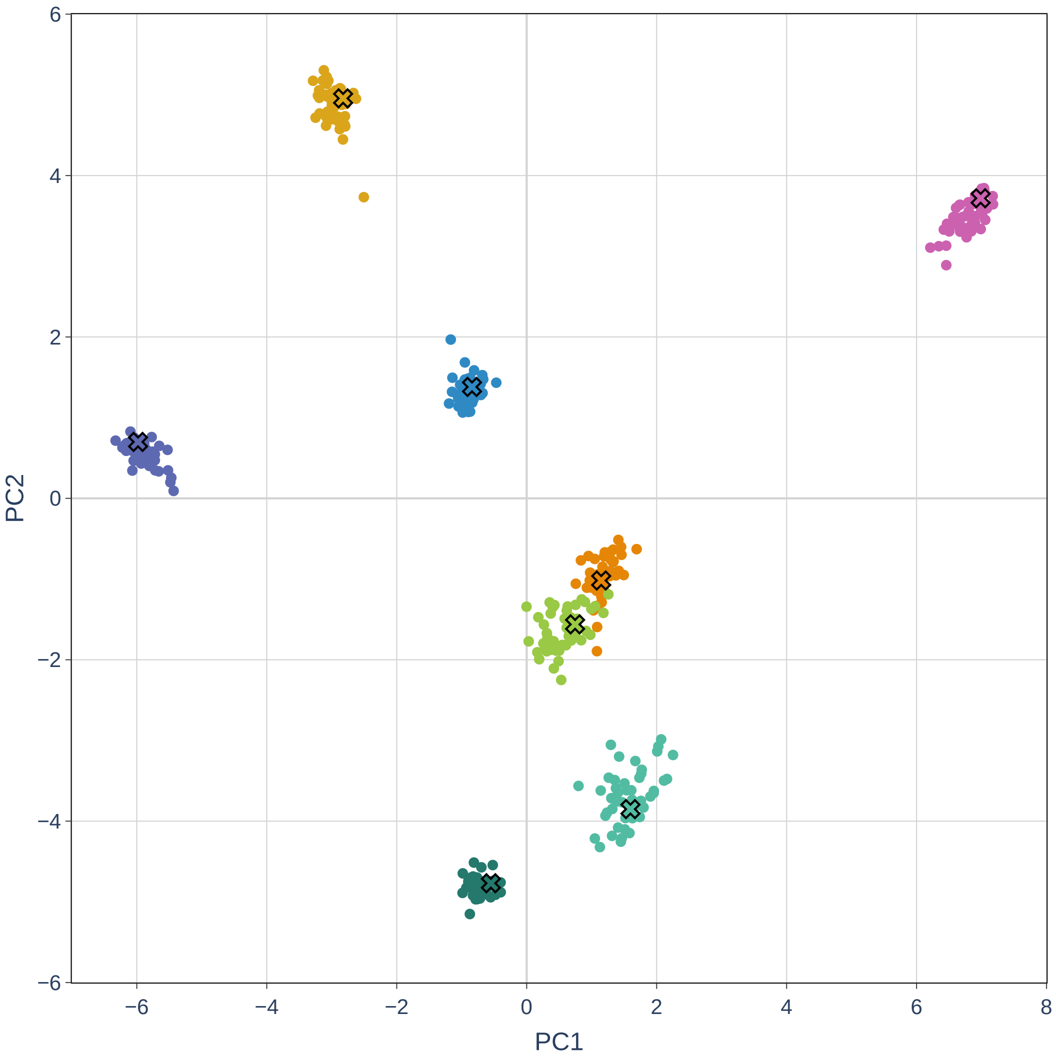


Figure 34: Epoch 28000

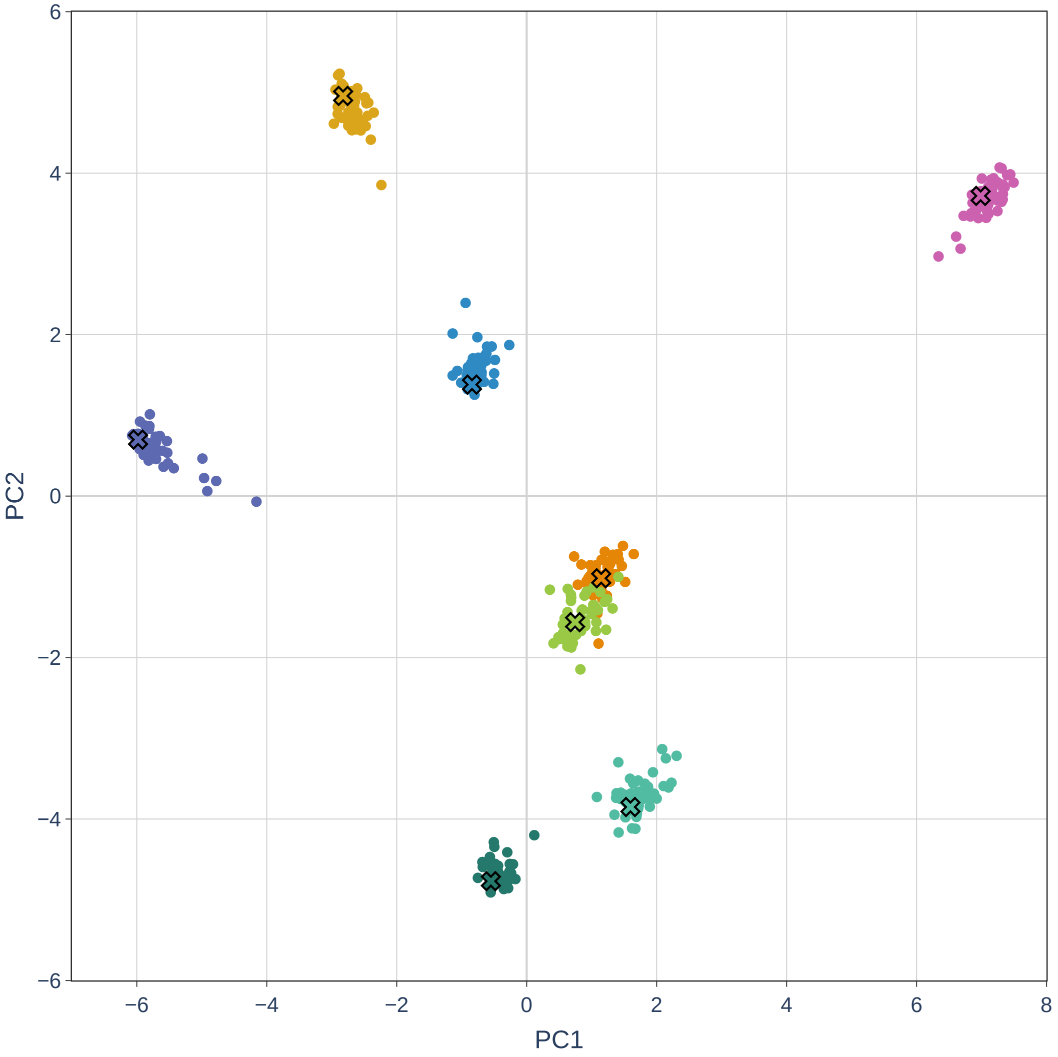


Figure 35: Epoch 29000

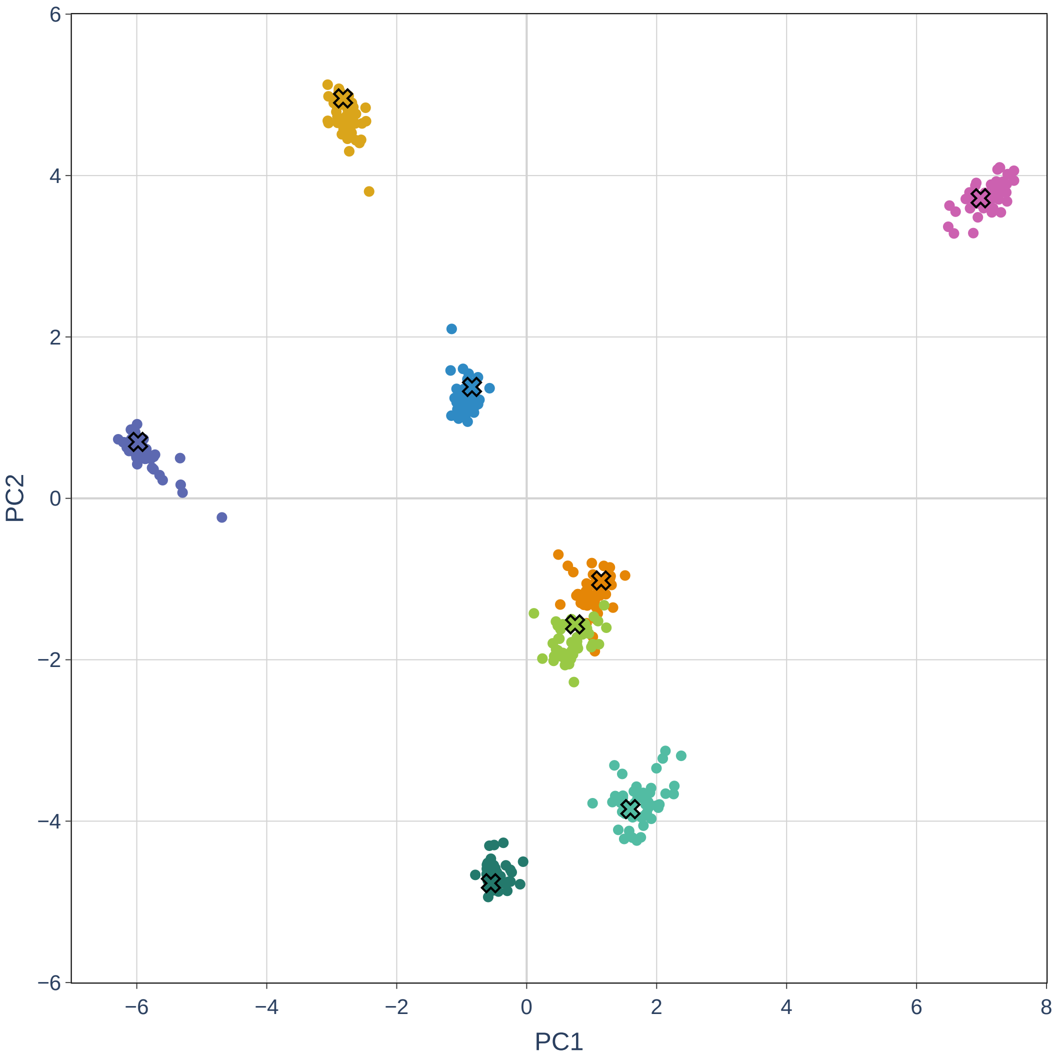


Figure 36: Epoch 30000

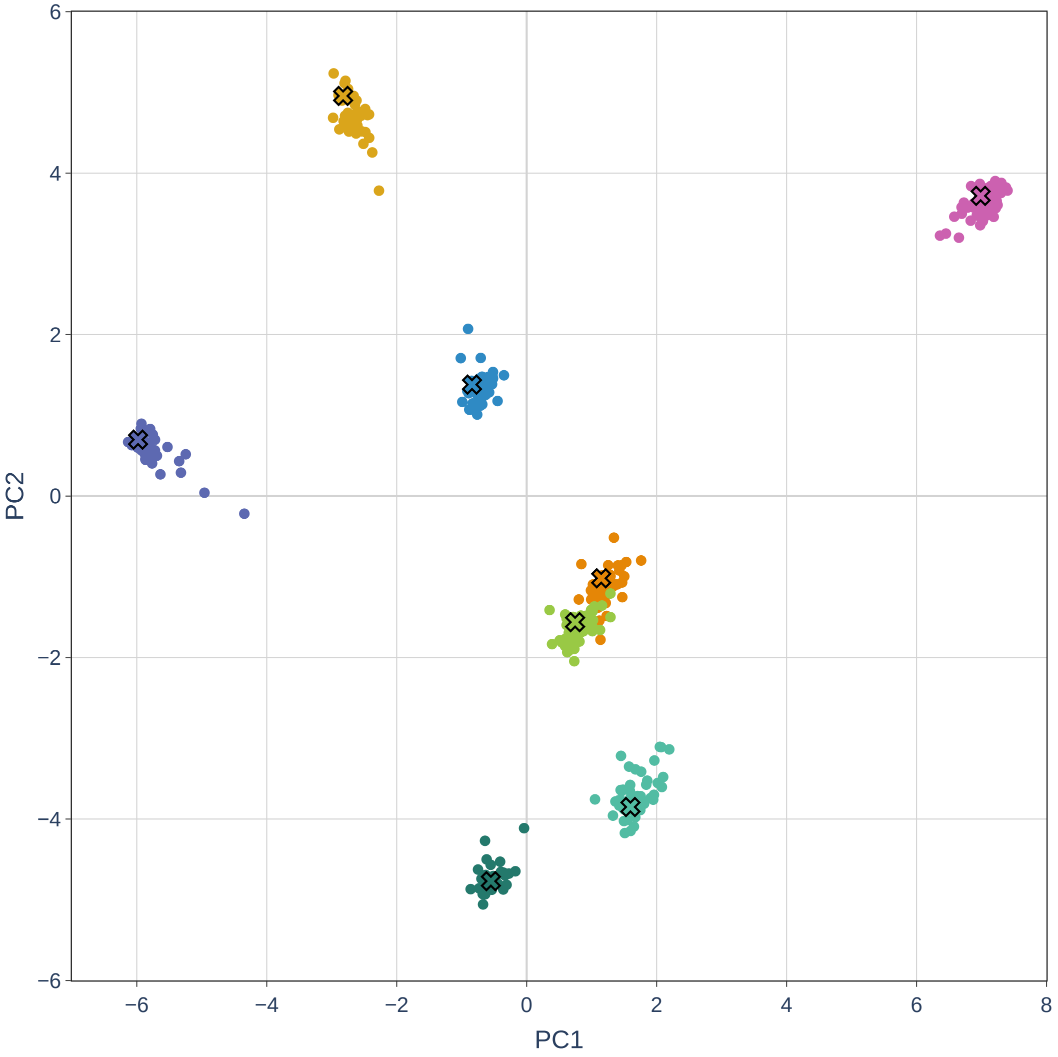


Figure 37: Epoch 31000

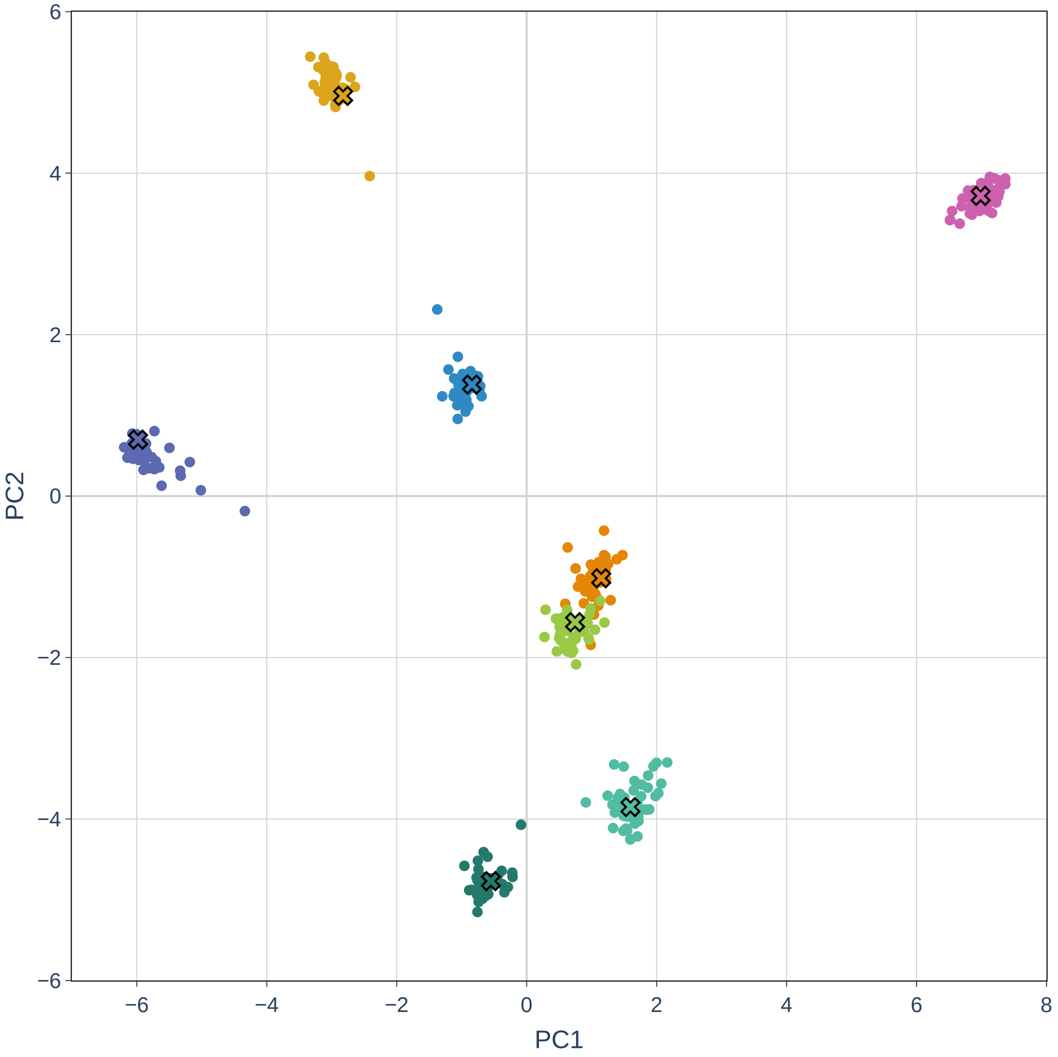


Figure 38: Epoch 32000

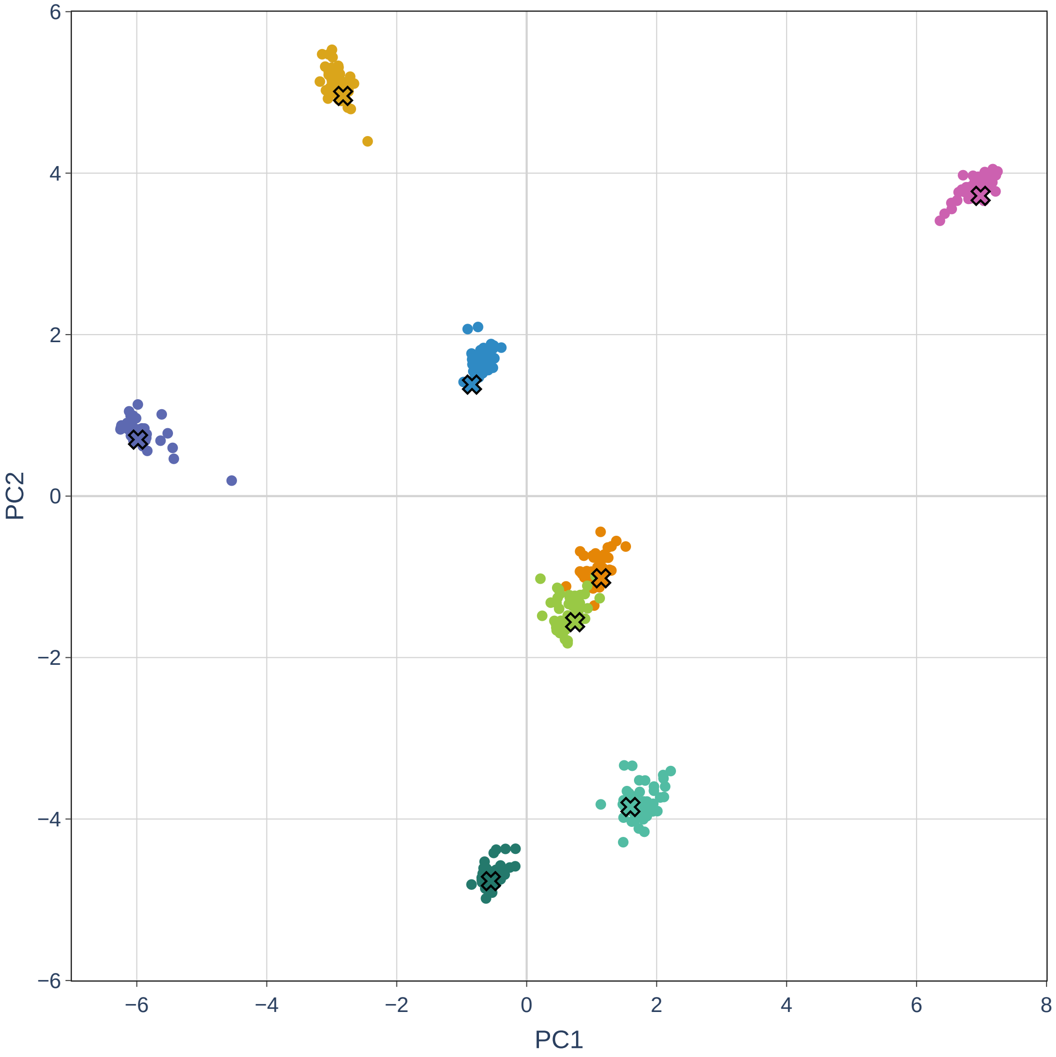


Figure 39: Epoch 33000

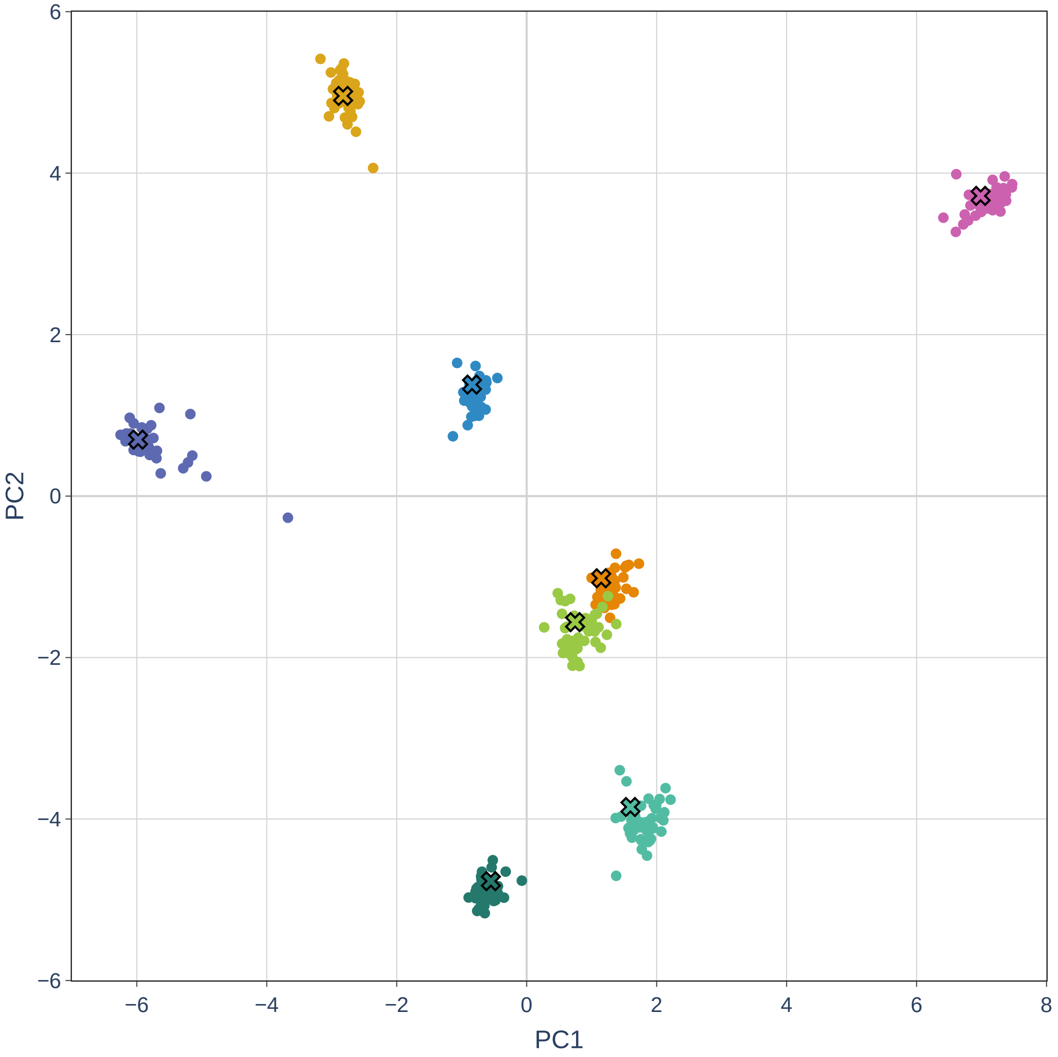


Figure 40: Epoch 34000

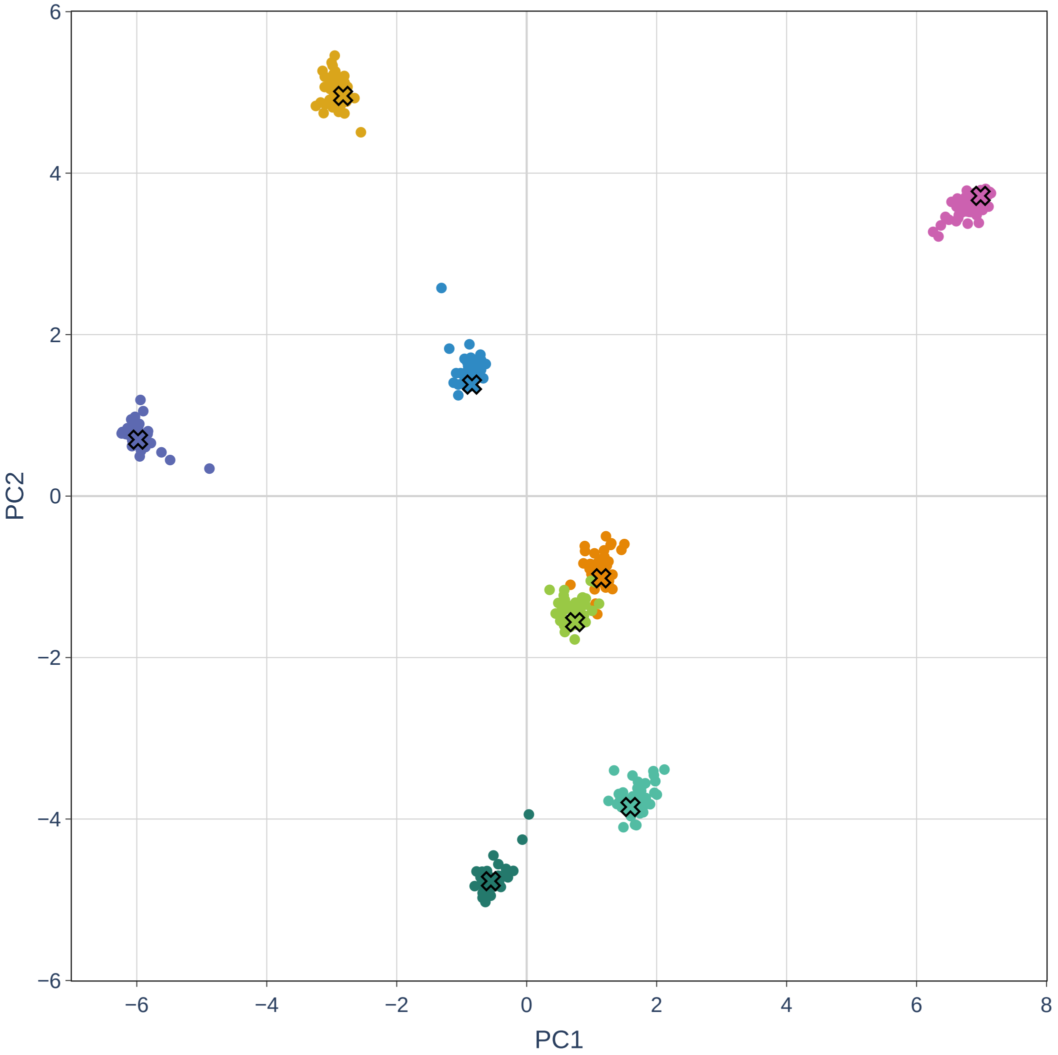


Figure 41: Epoch 35000

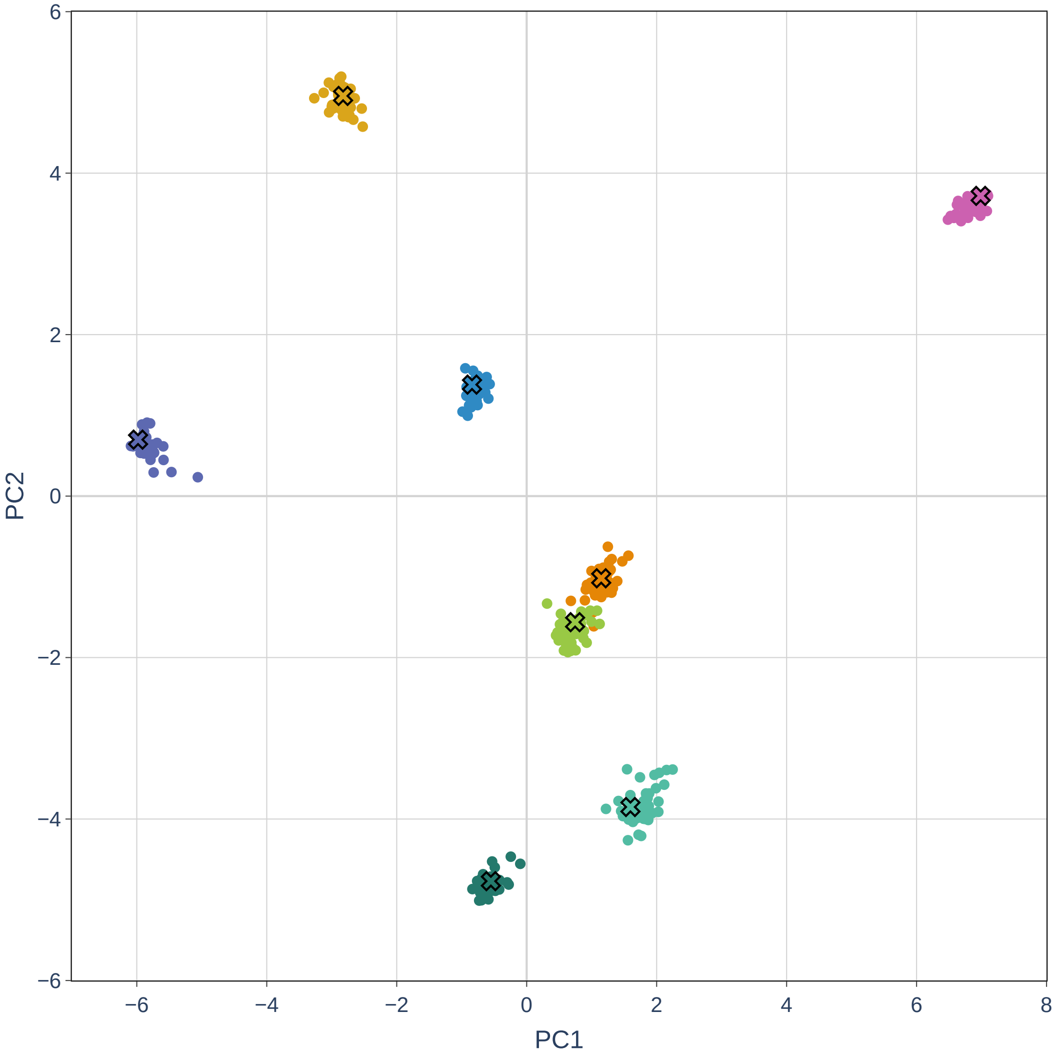


Figure 42: Epoch 36000