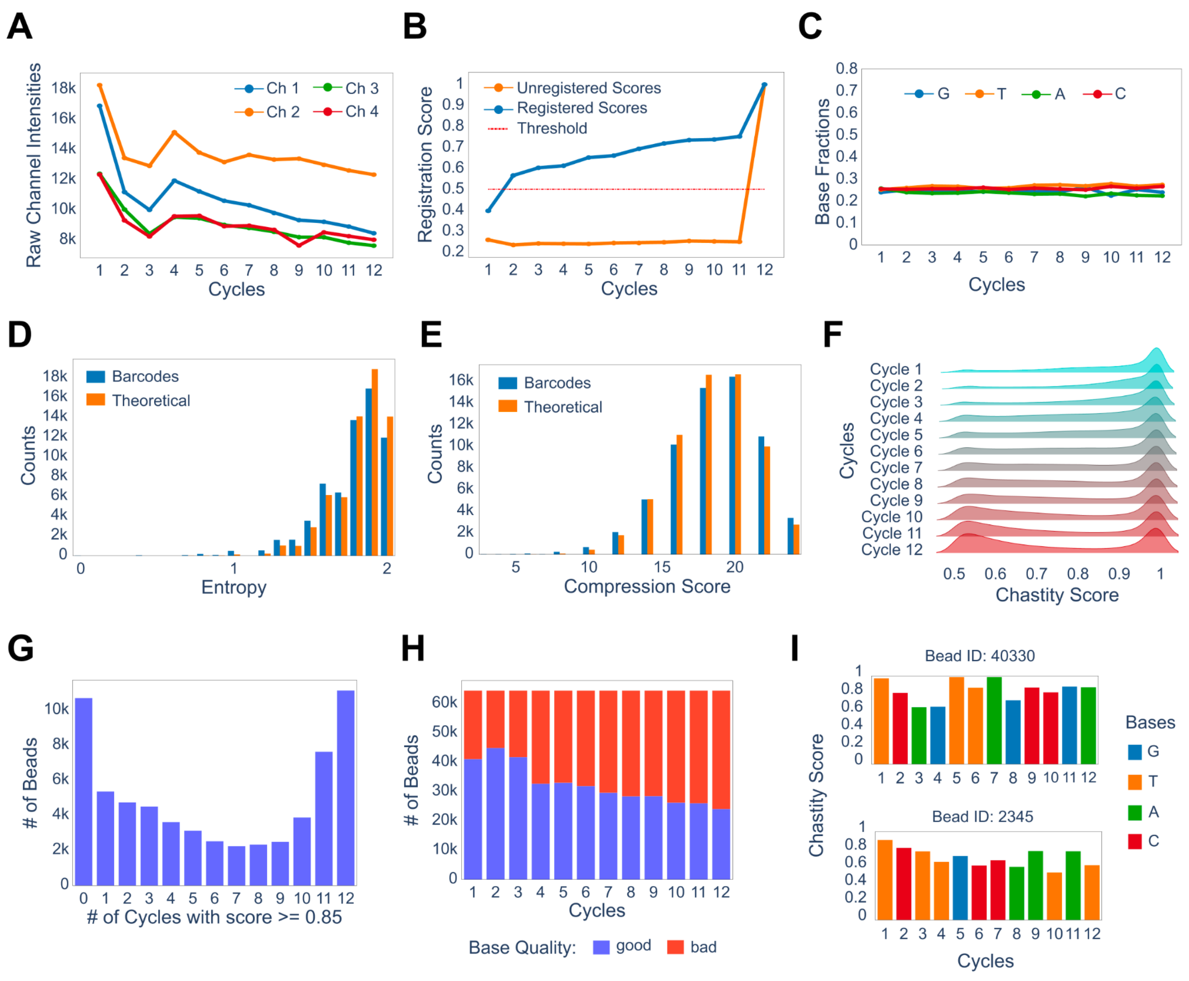
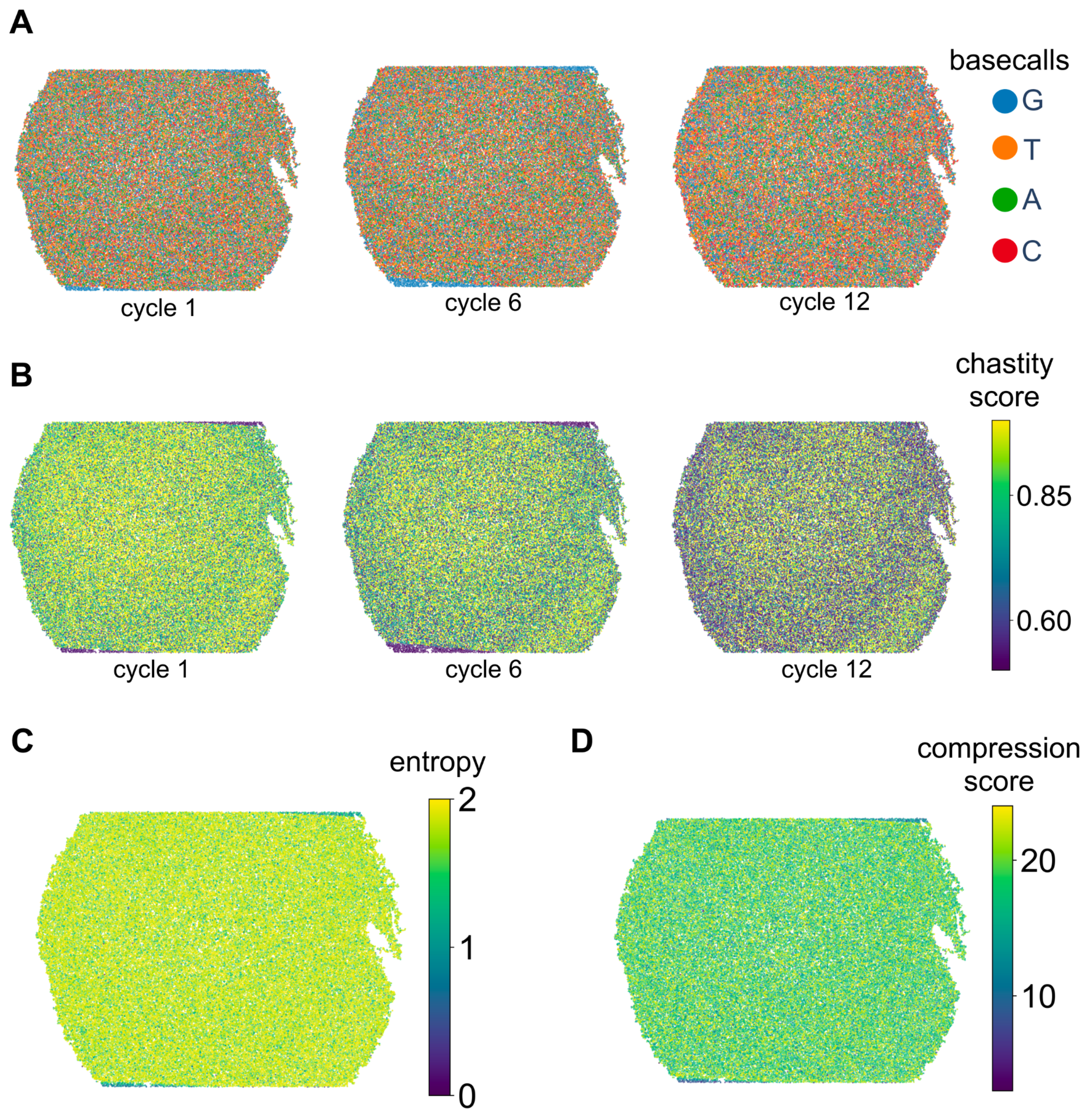
Supplementary Figures

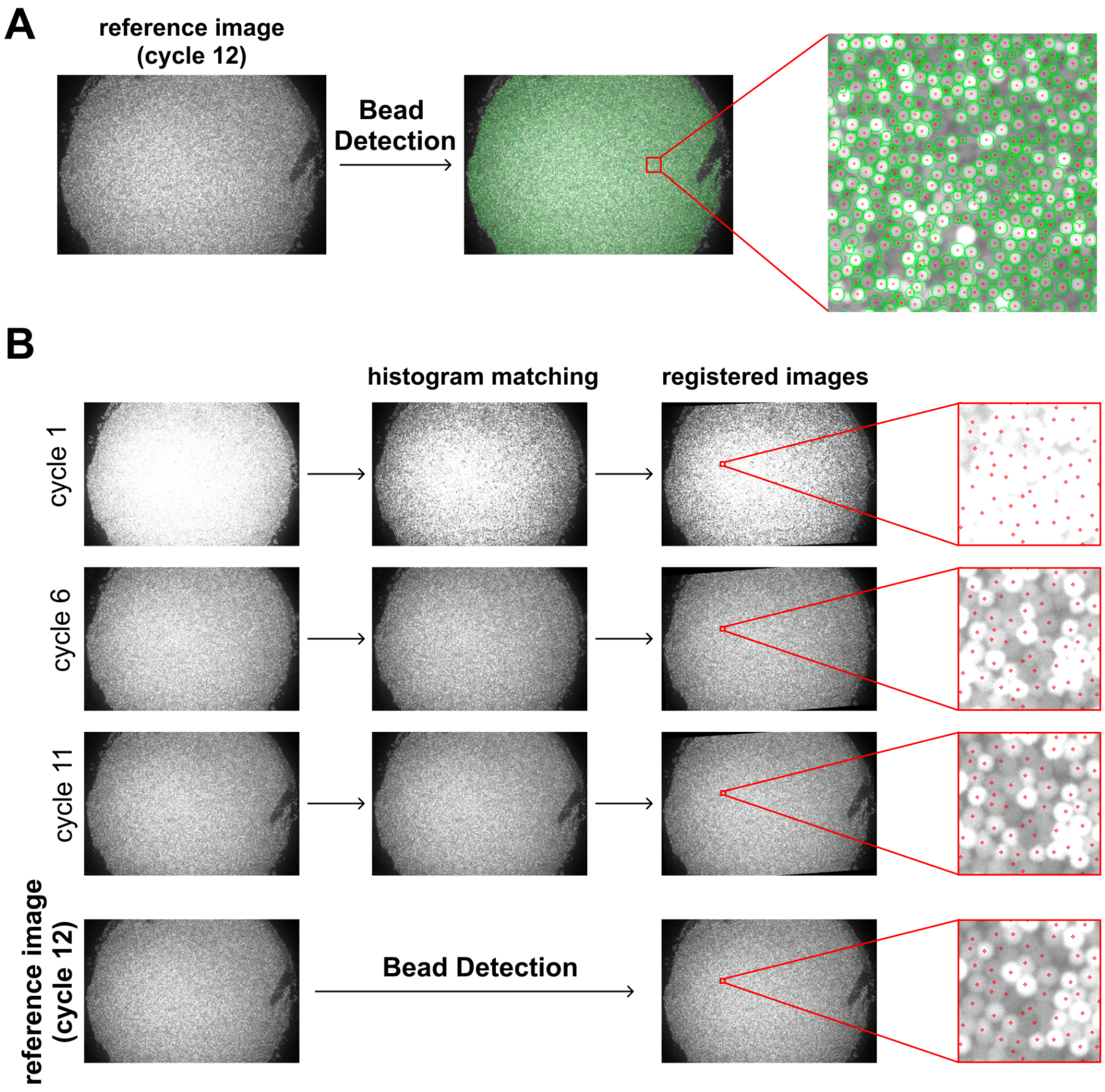
Sup. Fig. 1: Example of QC Plots shown for P4. **a,** Raw channel intensities across cycles. **b,** Similarity scores for registered and unregistered cycle images. Cycle 12 exhibits a perfect score, as previous cycles are registered to that. The dashed line at 0.5 denotes an empirical threshold. **c,** Called base fractions per cycle. **d,** Compression score distribution of the barcodes along with the expected distribution. **e,** Entropy score distribution of the barcodes along with the expected distribution. **f,** Basecalling chastity score distribution per cycle. **g,** Histogram of the beads that has n cycles higher than the threshold. **h,** Histogram of the base positions where a base is called with a score lower (red) or higher (blue) than the threshold. **i,** Scores per cycle for two randomly chosen beads.



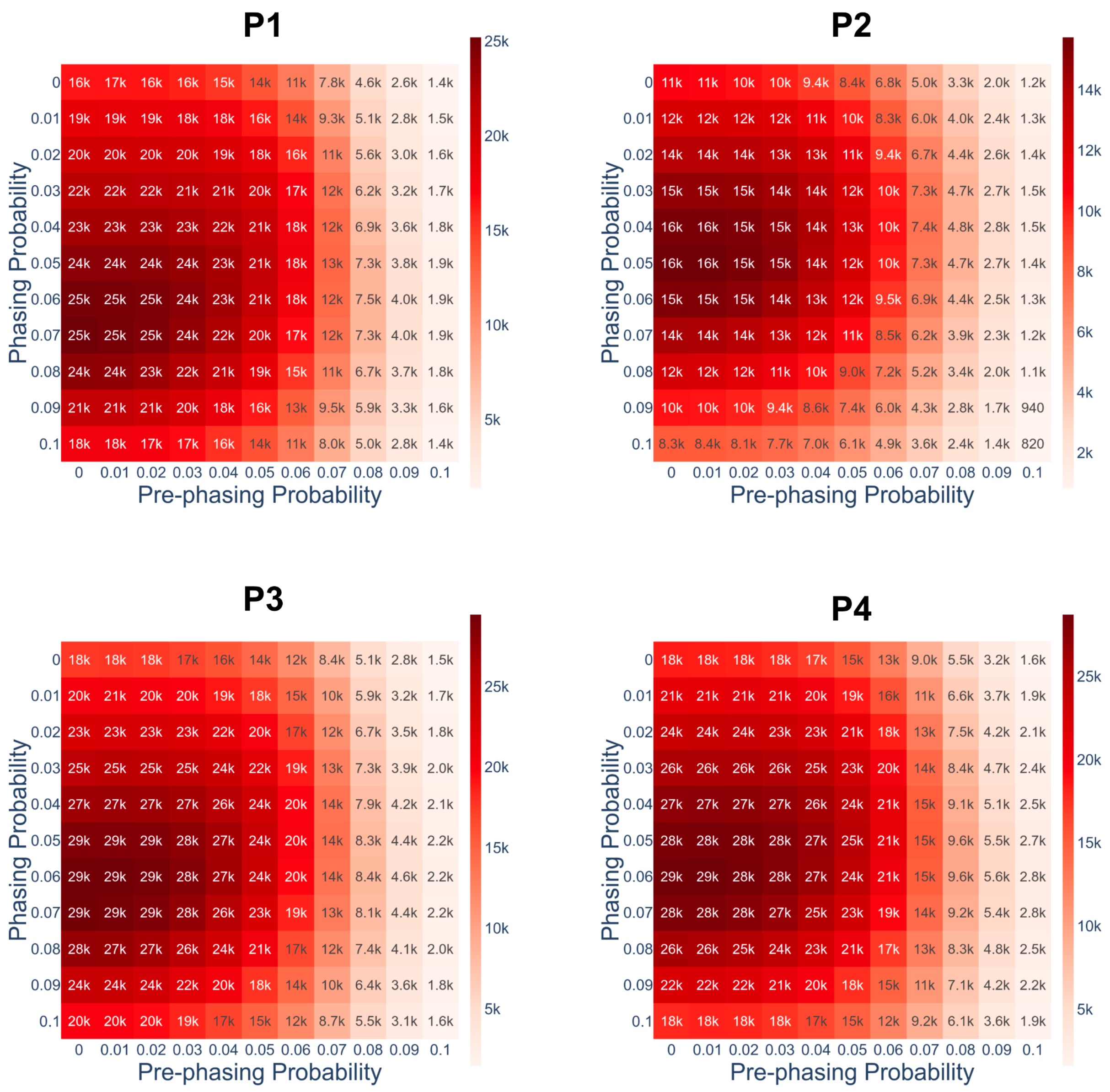
Sup. Fig. 2: Example of QC Sheet Plots shown for puck P4. **a,** Called bases in space for every cycle. Here shown cycles 1, 6 and 12. **b,** Chastity scores in space for every cycle. Here shown cycles 1, 6 and 12. **c,** Entropy score distribution in space. Higher values correspond to higher barcode complexity. **d,** Compression score distribution in space. Higher values correspond to higher barcode complexity.



Sup. Fig. 3: Registration and bead detection examples shown for P4. **a,** Beads are detected from the reference image which is the overlay of all channel intensities of the last cycle. **b,** Histogram matching and image registration examples shown for cycles 1, 6, 11 and 12.



Sup. Fig. 4: Heatmaps showing the number of matches between the optically decoded and Illumina sequenced barcodes as a function of phasing and prephasing probabilities for the pucks P1-P4. Moderate phasing and no pre-phasing effects are observed across all pucks.



P1

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	16k	17k	16k	16k	15k	14k	11k	7.8k	4.6k	2.6k	1.4k
0.01	19k	19k	19k	18k	18k	16k	14k	9.3k	5.1k	2.8k	1.5k
0.02	20k	20k	20k	20k	19k	18k	16k	11k	5.6k	3.0k	1.6k
0.03	22k	22k	22k	21k	21k	20k	17k	12k	6.2k	3.2k	1.7k
0.04	23k	23k	23k	23k	22k	21k	18k	12k	6.9k	3.6k	1.8k
0.05	24k	24k	24k	24k	23k	21k	18k	13k	7.3k	3.8k	1.9k
0.06	25k	25k	25k	24k	23k	21k	18k	12k	7.5k	4.0k	1.9k
0.07	25k	25k	25k	24k	22k	20k	17k	12k	7.3k	4.0k	1.9k
0.08	24k	24k	23k	22k	21k	19k	15k	11k	6.7k	3.7k	1.8k
0.09	21k	21k	21k	20k	18k	16k	13k	9.5k	5.9k	3.3k	1.6k
0.1	18k	18k	17k	17k	16k	14k	11k	8.0k	5.0k	2.8k	1.4k

P2

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	11k	11k	10k	10k	9.4k	8.4k	6.8k	5.0k	3.3k	2.0k	1.2k
0.01	12k	12k	12k	12k	11k	10k	8.3k	6.0k	4.0k	2.4k	1.3k
0.02	14k	14k	14k	13k	13k	11k	9.4k	6.7k	4.4k	2.6k	1.4k
0.03	15k	15k	15k	14k	14k	12k	10k	7.3k	4.7k	2.7k	1.5k
0.04	16k	16k	15k	15k	14k	13k	10k	7.4k	4.8k	2.8k	1.5k
0.05	16k	16k	15k	15k	14k	12k	10k	7.3k	4.7k	2.7k	1.4k
0.06	15k	15k	15k	14k	13k	12k	9.5k	6.9k	4.4k	2.5k	1.3k
0.07	14k	14k	14k	13k	12k	11k	8.5k	6.2k	3.9k	2.3k	1.2k
0.08	12k	12k	12k	11k	10k	9.0k	7.2k	5.2k	3.4k	2.0k	1.1k
0.09	10k	10k	10k	9.4k	8.6k	7.4k	6.0k	4.3k	2.8k	1.7k	940
0.1	8.3k	8.4k	8.1k	7.7k	7.0k	6.1k	4.9k	3.6k	2.4k	1.4k	820

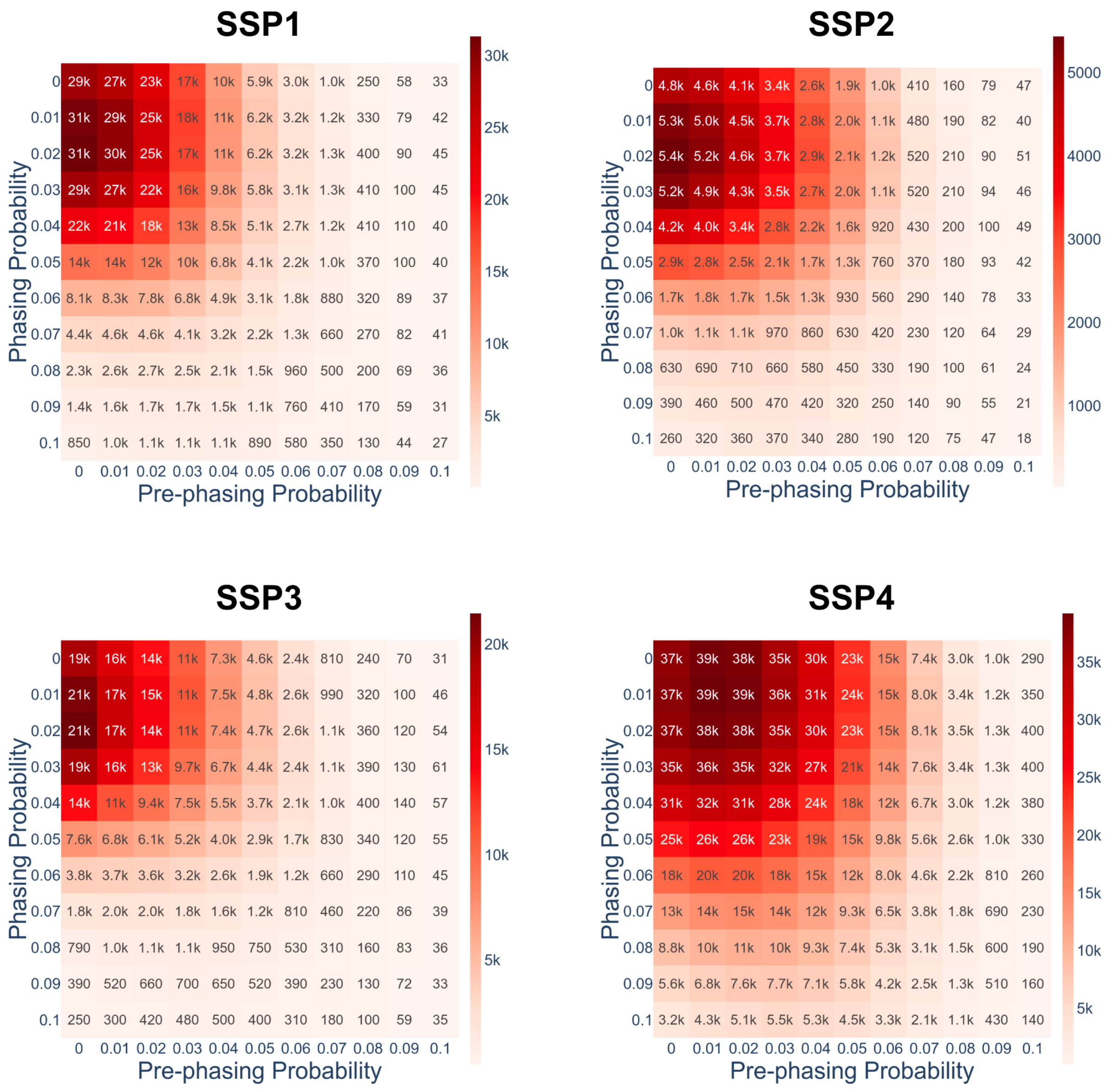
P3

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	18k	18k	18k	17k	16k	14k	12k	8.4k	5.1k	2.8k	1.5k
0.01	20k	21k	20k	20k	19k	18k	15k	10k	5.9k	3.2k	1.7k
0.02	23k	23k	23k	23k	22k	20k	17k	12k	6.7k	3.5k	1.8k
0.03	25k	25k	25k	25k	24k	22k	19k	13k	7.3k	3.9k	2.0k
0.04	27k	27k	27k	27k	26k	24k	20k	14k	7.9k	4.2k	2.1k
0.05	29k	29k	29k	28k	27k	24k	20k	14k	8.3k	4.4k	2.2k
0.06	29k	29k	29k	28k	27k	24k	20k	14k	8.4k	4.6k	2.2k
0.07	29k	29k	29k	28k	26k	23k	19k	13k	8.1k	4.4k	2.2k
0.08	28k	27k	27k	26k	24k	21k	17k	12k	7.4k	4.1k	2.0k
0.09	24k	24k	24k	22k	20k	18k	14k	10k	6.4k	3.6k	1.8k
0.1	20k	20k	20k	19k	17k	15k	12k	8.7k	5.5k	3.1k	1.6k

P4

	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	18k	18k	18k	18k	17k	15k	13k	9.0k	5.5k	3.2k	1.6k
0.01	21k	21k	21k	21k	20k	19k	16k	11k	6.6k	3.7k	1.9k
0.02	24k	24k	24k	23k	23k	21k	18k	13k	7.5k	4.2k	2.1k
0.03	26k	26k	26k	26k	25k	23k	20k	14k	8.4k	4.7k	2.4k
0.04	27k	27k	27k	27k	26k	24k	21k	15k	9.1k	5.1k	2.5k
0.05	28k	28k	28k	28k	27k	25k	21k	15k	9.6k	5.5k	2.7k
0.06	29k	29k	28k	28k	27k	24k	21k	15k	9.6k	5.6k	2.8k
0.07	28k	28k	28k	27k	25k	23k	19k	14k	9.2k	5.4k	2.8k
0.08	26k	26k	25k	24k	23k	21k	17k	13k	8.3k	4.8k	2.5k
0.09	22k	22k	22k	21k	20k	18k	15k	11k	7.1k	4.2k	2.2k
0.1	18k	18k	18k	18k	17k	15k	12k	9.2k	6.1k	3.6k	1.9k

Sup. Fig. 5: Heatmaps showing the number of matches between the optically decoded and Illumina sequenced barcodes as a function of phasing and prephasing probabilities for the Slide-Seq and Slide-SeqV2 pucks. Little-to-no phasing and pre-phasing effects are observed across all pucks.



SSP1

Phasing \ Pre-phasing	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	29k	27k	23k	17k	10k	5.9k	3.0k	1.0k	250	58	33
0.01	31k	29k	25k	18k	11k	6.2k	3.2k	1.2k	330	79	42
0.02	31k	30k	25k	17k	11k	6.2k	3.2k	1.3k	400	90	45
0.03	29k	27k	22k	16k	9.8k	5.8k	3.1k	1.3k	410	100	45
0.04	22k	21k	18k	13k	8.5k	5.1k	2.7k	1.2k	410	110	40
0.05	14k	14k	12k	10k	6.8k	4.1k	2.2k	1.0k	370	100	40
0.06	8.1k	8.3k	7.8k	6.8k	4.9k	3.1k	1.8k	880	320	89	37
0.07	4.4k	4.6k	4.6k	4.1k	3.2k	2.2k	1.3k	660	270	82	41
0.08	2.3k	2.6k	2.7k	2.5k	2.1k	1.5k	960	500	200	69	36
0.09	1.4k	1.6k	1.7k	1.7k	1.5k	1.1k	760	410	170	59	31
0.1	850	1.0k	1.1k	1.1k	1.1k	890	580	350	130	44	27

SSP2

Phasing \ Pre-phasing	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	4.8k	4.6k	4.1k	3.4k	2.6k	1.9k	1.0k	410	160	79	47
0.01	5.3k	5.0k	4.5k	3.7k	2.8k	2.0k	1.1k	480	190	82	40
0.02	5.4k	5.2k	4.6k	3.7k	2.9k	2.1k	1.2k	520	210	90	51
0.03	5.2k	4.9k	4.3k	3.5k	2.7k	2.0k	1.1k	520	210	94	46
0.04	4.2k	4.0k	3.4k	2.8k	2.2k	1.6k	920	430	200	100	49
0.05	2.9k	2.8k	2.5k	2.1k	1.7k	1.3k	760	370	180	93	42
0.06	1.7k	1.8k	1.7k	1.5k	1.3k	930	560	290	140	78	33
0.07	1.0k	1.1k	1.1k	970	860	630	420	230	120	64	29
0.08	630	690	710	660	580	450	330	190	100	61	24
0.09	390	460	500	470	420	320	250	140	90	55	21
0.1	260	320	360	370	340	280	190	120	75	47	18

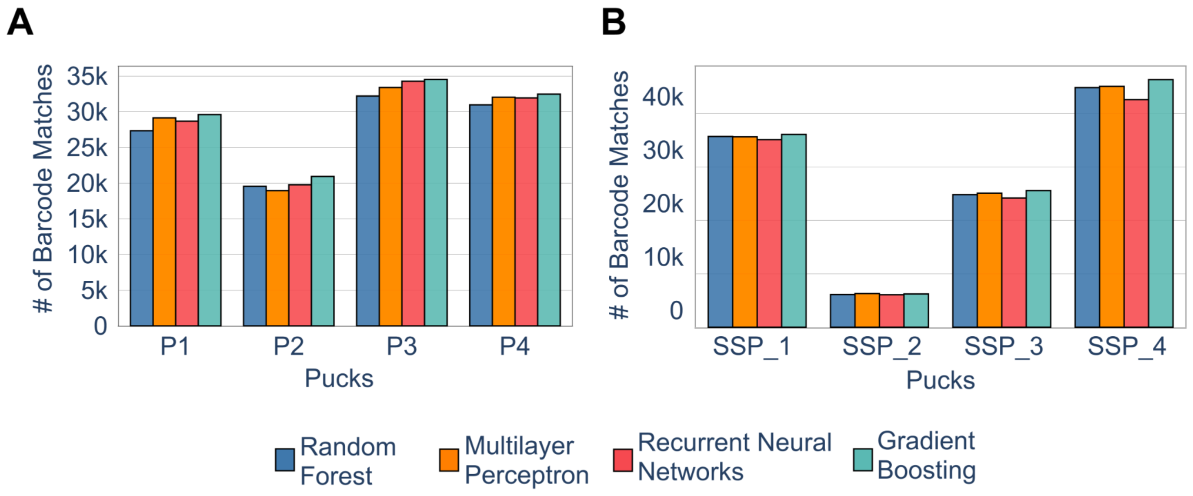
SSP3

Phasing \ Pre-phasing	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	19k	16k	14k	11k	7.3k	4.6k	2.4k	810	240	70	31
0.01	21k	17k	15k	11k	7.5k	4.8k	2.6k	990	320	100	46
0.02	21k	17k	14k	11k	7.4k	4.7k	2.6k	1.1k	360	120	54
0.03	19k	16k	13k	9.7k	6.7k	4.4k	2.4k	1.1k	390	130	61
0.04	14k	11k	9.4k	7.5k	5.5k	3.7k	2.1k	1.0k	400	140	57
0.05	7.6k	6.8k	6.1k	5.2k	4.0k	2.9k	1.7k	830	340	120	55
0.06	3.8k	3.7k	3.6k	3.2k	2.6k	1.9k	1.2k	660	290	110	45
0.07	1.8k	2.0k	2.0k	1.8k	1.6k	1.2k	810	460	220	86	39
0.08	790	1.0k	1.1k	1.1k	950	750	530	310	160	83	36
0.09	390	520	660	700	650	520	390	230	130	72	33
0.1	250	300	420	480	500	400	310	180	100	59	35

SSP4

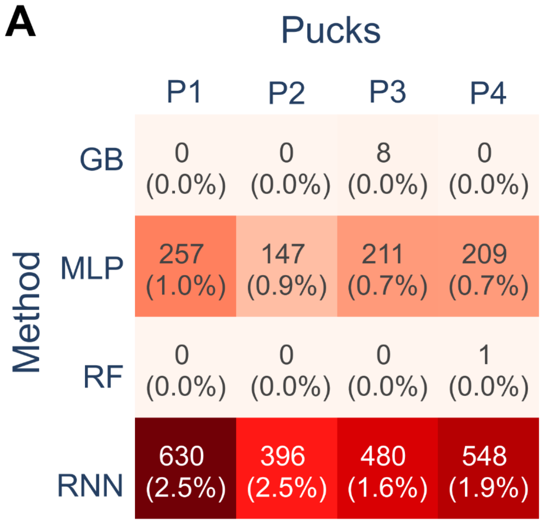
Phasing \ Pre-phasing	0	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0	37k	39k	38k	35k	30k	23k	15k	7.4k	3.0k	1.0k	290
0.01	37k	39k	39k	36k	31k	24k	15k	8.0k	3.4k	1.2k	350
0.02	37k	38k	38k	35k	30k	23k	15k	8.1k	3.5k	1.3k	400
0.03	35k	36k	35k	32k	27k	21k	14k	7.6k	3.4k	1.3k	400
0.04	31k	32k	31k	28k	24k	18k	12k	6.7k	3.0k	1.2k	380
0.05	25k	26k	26k	23k	19k	15k	9.8k	5.6k	2.6k	1.0k	330
0.06	18k	20k	20k	18k	15k	12k	8.0k	4.6k	2.2k	810	260
0.07	13k	14k	15k	14k	12k	9.3k	6.5k	3.8k	1.8k	690	230
0.08	8.8k	10k	11k	10k	9.3k	7.4k	5.3k	3.1k	1.5k	600	190
0.09	5.6k	6.8k	7.6k	7.7k	7.1k	5.8k	4.2k	2.5k	1.3k	510	160
0.1	3.2k	4.3k	5.1k	5.5k	5.3k	4.5k	3.3k	2.1k	1.1k	430	140

Sup. Fig. 6: Benchmarking the performance of various classifiers for the machine learning basecaller. **a,** Number of matches between the optically decoded and the Illumina sequenced barcodes for the in-house pucks and for four different machine learning classifiers. **b,** Same as in a, but for the Slide-Seq and Slide-SeqV2 pucks.



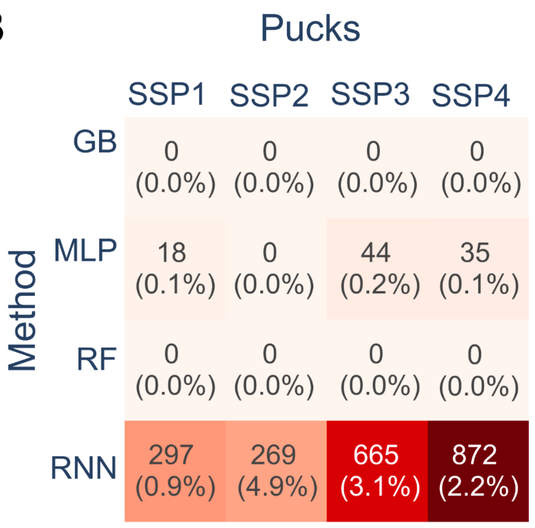
Sup. Fig. 7: Heatmaps for specificity analysis of machine learning models. Here, different algorithms are employed to predict the true barcodes of the training set (i.e. the barcodes that match with the Illumina sequences after phasing correction). The values show the absolute number and the percentage of misclassified barcodes a) for in-house pucks and b) for Slide-Seq and Slide-SeqV2 pucks. GB: Gradient Boosting; MLP: Multilayer Perceptron; RF: Random Forests; RNN: Recurrent Neural Networks

A



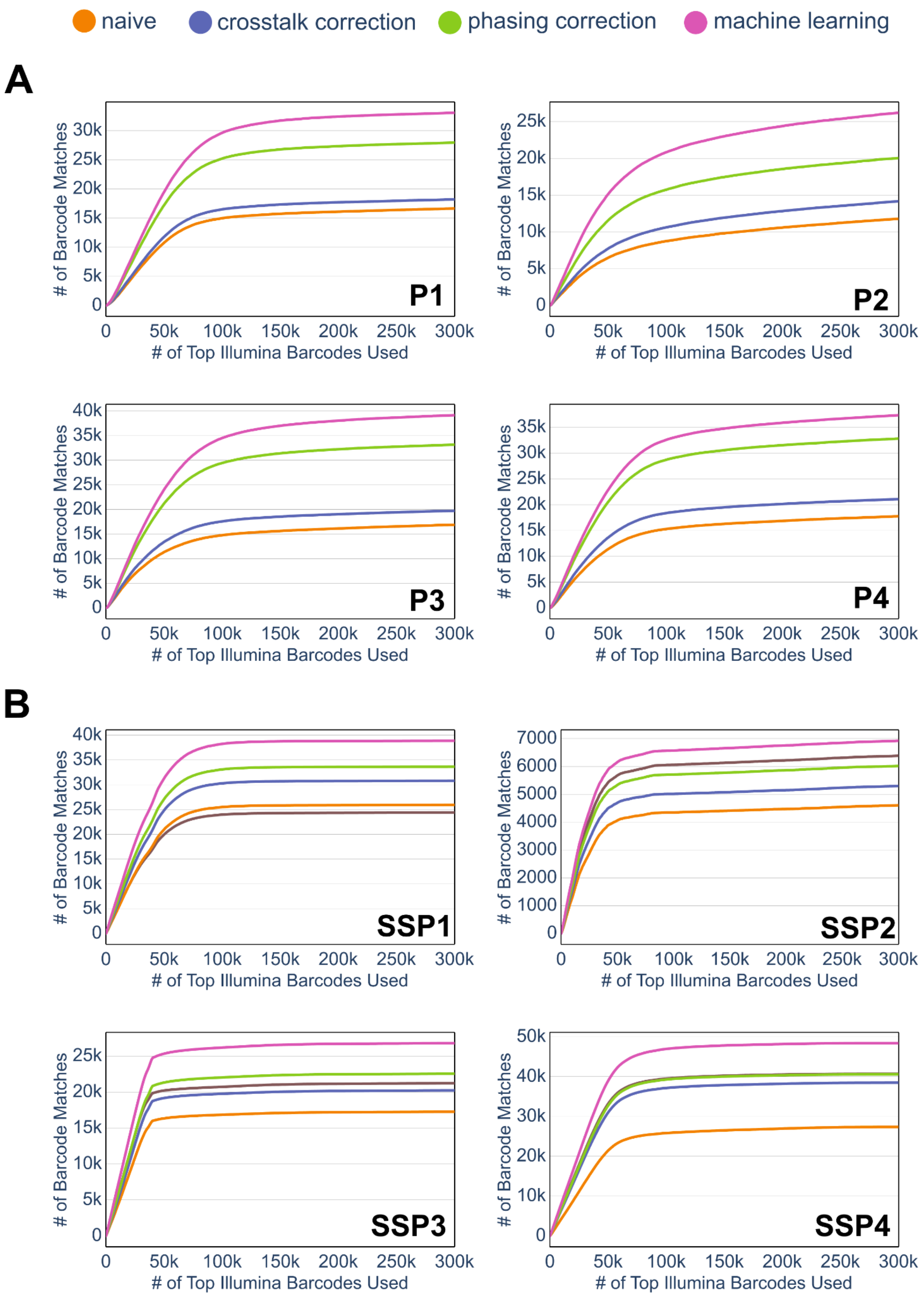
	P1	P2	P3	P4
GB	0 (0.0%)	0 (0.0%)	8 (0.0%)	0 (0.0%)
MLP	257 (1.0%)	147 (0.9%)	211 (0.7%)	209 (0.7%)
RF	0 (0.0%)	0 (0.0%)	0 (0.0%)	1 (0.0%)
RNN	630 (2.5%)	396 (2.5%)	480 (1.6%)	548 (1.9%)

B

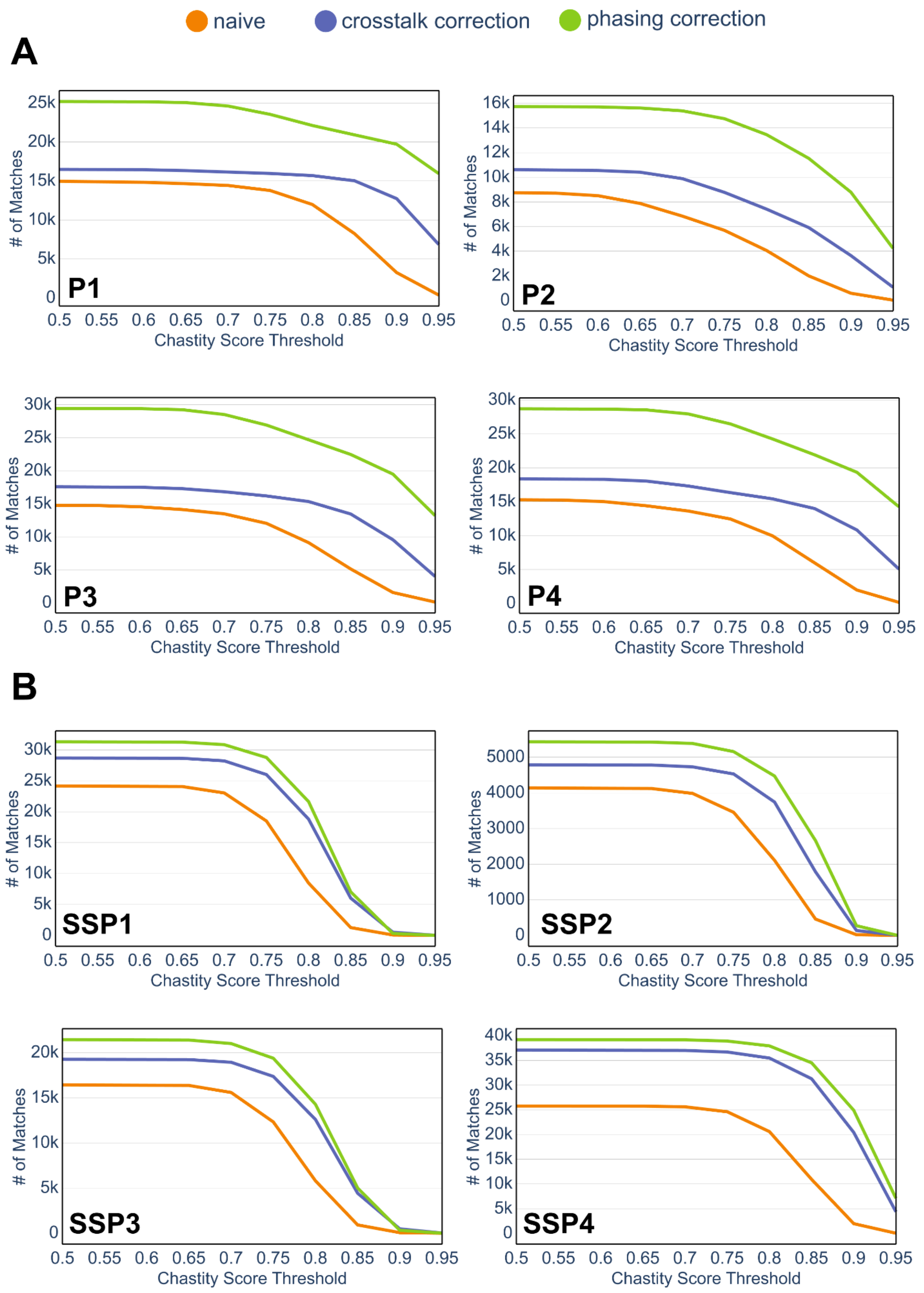


	SSP1	SSP2	SSP3	SSP4
GB	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
MLP	18 (0.1%)	0 (0.0%)	44 (0.2%)	35 (0.1%)
RF	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
RNN	297 (0.9%)	269 (4.9%)	665 (3.1%)	872 (2.2%)

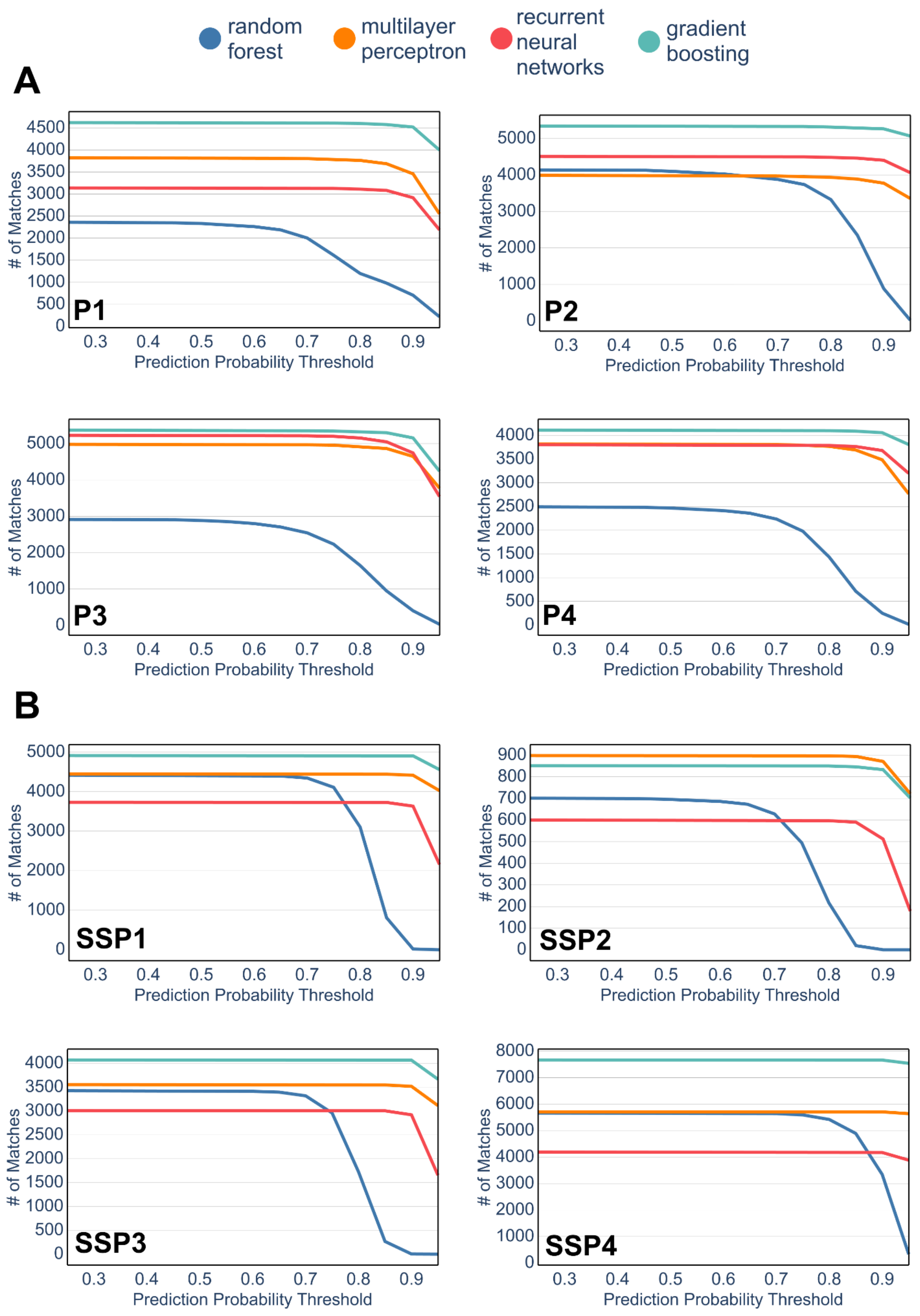
Sup. Fig. 8: Cumulative plots showing the number of matches between the optically decoded and Illumina sequenced barcodes as a function of the number of top Illumina barcodes used a) for in-house pucks and b) for Slide-Seq and Slide-SeqV2 pucks



Sup. Fig. 9: Number of matched barcodes after filtering them with a chastity score threshold for the basecalling methods. Chastity score used is the mean score of the barcode's cycles a) for in-house pucks and b) for Slide-Seq and Slide-SeqV2 pucks



Sup. Fig. 10: Number of matched barcodes after filtering them with the prediction score threshold for the machine learning methods. Prediction score is the probability of the called base by the model and here the mean score of the barcode's cycles is used a) for in-house pucks and b) for Slide-Seq and Slide-SeqV2 pucks



Supplementary Methods

Image Processing

Puck images generated through the in-house platform consist of 6-channel images for every cycle, where 4 channels were retained for further processing and the other two were ignored since they were only used for internal microscopy setup control purposes. Pucks from Slide-Seq and Slide-SeqV2 datasets are 4-channel images which were directly used for further processing. For Slide-Seq, a TIFF file is provided per channel and we combined the channels to create a multipage TIFF image for each cycle. For Slide-SeqV2 raw microscopy data we used Puckcaller (https://github.com/MacoskoLab/PuckCaller) to generate the stitched TIFF files. After loading the images, an overlay image for each cycle was generated by summing up the intensities of all channels and was subsequently used for bead detection and image registration. Image processing operations described below were implemented with the OpenCV (v4.2.0) and Skimage (v0.18.3) packages.

Bead Detection

Beads were detected from the last cycle’s overlay image. First, a median blur (3x3) for noise removal and contrast limited adaptive histogram equalisation (CLAHE) with 64x64 tiles were applied to the overlay image. Then, Optocoder used the Hough Circle Transform OpenCV implementation to detect beads. By default, Optocoder searches for beads with a radius range of 4-10 pixels. These parameter and edge detection thresholds for Hough Transform can be modified through the bead detection parameters as necessary. For the results presented in this paper, we used the same parameters in all runs for both in-house and Slide-Seq datasets. Furthermore, puck borders were detected by searching for the biggest connected component in the image after applying a Gaussian blur. Detected beads outside of the main puck area were ignored.

Image Registration

Every image was registered to the final cycle to correct misalignments between cycles. First, a histogram matching step was applied. This was done via calculating the cumulative distribution function for both the image-to-be-registered and the reference and consecutively matching them. Image registration was done via the Enhanced Cross Correlation Maximization (1) algorithm implementation in OpenCV with a Euclidean motion model. Registration was done with a pyramidal scheme where the image was downsampled eight times to speed up the registration process and improve the accuracy by aligning the image in different resolutions, starting from the coarse to the finer resolutions. Warping parameters were acquired from the registration of the overlay images and they were consecutively applied to the channel images.

Background Correction

For every cycle and every channel, the background of the image was calculated by using a morphological opening operation with a 64x64 rectangular kernel. This background was subtracted from the respective channel before correcting for crosstalk and phasing effects.

Basecalling

Input to basecalling is the detected beads $Beads = (B_1, ..., B_N)$ where $B_i \in R^{M \times 4}$ is bead i, N is the number of detected beads and M is the number of cycles. Every bead has a 4-dimensional intensity vector for every sequencing cycle.

Calculating the Crosstalk Matrix

For crosstalk correction, Optocoder uses the iterative correction method from (2). Crosstalk Matrix M is a 4x4 matrix which represents the interaction between channels. We calculate the crosstalk for all pairs of channels such that each corresponds to one element in the crosstalk matrix. This matrix is calculated from the bead intensities of the first cycle.
First, M is initiated as an identity matrix. Then, the crosstalk for every pair of channels is calculated as below:
For a given pair of channels (ch_a, ch_b):

1. The beads that have intensities between the 60th and 99th quantiles for the first component are chosen. $Beads^* = \{b_i \mid Q_A(0.60) < b_{i,1,a} < Q_A(0.99)\}$ where A is the intensities of all beads in the first cycle for the first component channel and $b_{i,1,a}$ is the intensity of *channel a* of bead *i* at cycle 1.
2. Beads in $Beads^*$ are binned such that every bin includes ~10 beads.
3. A linear regression model is fitted to $Beads^*$ and the slope of this model is consecutively used to fill the crosstalk matrix entry for the respective pair.

The inverse of the crosstalk matrix is multiplied with the intensity data $Beads$ to calculate the corrected intensities for all beads. The whole process is repeated to update the crosstalk Matrix M, until either 15 iterations are reached or the maximum slope is lower than 0.05.

Generating the Phasing Matrix

Phasing matrix is defined with two values where p is the probability or expected amount of phasing and q is the expected amount of prephasing. Phasing matrix P is initialised as an identity matrix of size (cycles x cycles) and is filled via a dynamic programming approach similar to (3).

$$P(i,j) = \begin{cases} p \cdot P(i-1,j) & j = 0 \\ p \cdot P(i-1,j) + (1-p-q) \cdot P(i-1,j-1) & j = 1 \\ p \cdot P(i-1,j) + (1-p-q) \cdot p(i-1,j-1) + q \cdot P(i-1,j-2) & \text{otherwise} \end{cases}$$

The values p and q can be selected by the user. Optocoder implements 0.07 phasing and 0.01 prephasing values by default.

Basecalling

Crosstalk and Phasing Correction Model

The crosstalk and phasing correction model is defined as:

$$B_i = C S_i P,$$

where $B_i \in R^{M\,x\,4}$ is a matrix containing the observed intensities of bead i, $C \in R^{4\,x\,4}$ is the crosstalk matrix, $S_i \in R^{M\,x\,4}$ are the true intensities of bead i, and $P \in R^{M\,x\,M}$ is the phasing matrix, where M is the number of cycles.

First, we calculate the crosstalk matrix and generate the phasing matrix as described above. Then, the inverse of the Kronecker product of the two matrices $(C \otimes P)^{-1}$ is calculated. This resulting correction matrix is multiplied with the bead intensities to acquire the corrected intensities.

$$S_i = (C \otimes P)^{-1} B_i \ . \qquad (1)$$

For the baseline basecallers, we use two different intensity profiles. For *naive basecalling,* we used the detected bead intensities after background correction without any crosstalk and phasing correction as:

$$S_i = B_i \ . \qquad (2)$$

For *crosstalk-only correction basecalling*, we first calculate the crosstalk matrix C as described above and the phasing matrix P is taken to be the Identity matrix.

$$S_i = (C \otimes I)^{-1} B_i \ . \qquad (3)$$

For the combined correction model, Phasing Matrix P is computed as explained above with the default parameters and applied to (1).

Scaling and Basecalling

Before basecalling, we first scaled the intensity values for every channel using a Robust Scaler, defined as:

$$\frac{ch_{kl} - Q_1(ch_{kl})}{Q_3(ch_{kl}) - Q_1(ch_{kl})},$$

where ch_{kl} are the intensities after corrections of all beads, S, for channel k in cycle l, and $Q3$ and $Q1$ are the quartiles 3 and 1, respectively. Finally, we applied a SoftMax function to the final intensities for each bead. Finally, the bases are called for each bead and each cycle by taking the maximum intensity channel.

Phasing Parameter Search

To find the phasing and prephasing parameters that maximise the number of matches, we search for every pair of phasing and prephasing values between 0.0 and 1.0 with 0.01 step size. We calculated the number of matches for every pair and the parameter set with the maximum number of matches is selected. Then we run the whole pipeline again with the obtained optimised parameters.

Quality Control Measures

Compression Score

Compression score for every barcode is calculated by computing the length of a compressed barcode. A barcode is compressed by computing the number of consecutive bases that are identical. For example, a repetitive, low complexity barcode such as “GGGGGGAAAAAA” would be compressed as “G6A6” while a complex barcode such as “GTACACATGCAC” would be compressed as “G1T1A1C1A1C1A1T1G1C1A1C1”. In the QC plots, compression score distribution of a puck is compared against a theoretical distribution that is expected. The theoretical distribution is computed by calculating the compression score for a set of randomly generated barcodes of the same size with BC.

Shannon Entropy

The Shannon entropy for every barcode is calculated by:

$$H_{bc_i} = -\sum_{n \in bc_i} f(n, bc_i)^* \log_2(f(n, bc_i)),$$

where $f(n,\ bc_i)$ is the relative frequency of barcode bc_i for $bc_i \in BC$. In the QC plots, entropy score distribution of a puck is compared against a theoretical distribution that is expected. The theoretical distribution is computed by calculating the Shannon entropy for a set of randomly generated barcodes of the same size with BC.

Chastity Score

We measure our base calling confidence by computing the chastity

$$C^{pq} = \frac{I^{pq}_{(n)}}{I^{pq}_{(n)} + I^{pq}_{(n-1)}} ,$$

where $I^{pq}_{(n)}$ and $I^{pq}_{(n-1)}$ are the intensities of the channels with the highest and the second highest values for bead p in cycle q.

Registration Quality

Alignment score between two cycles is calculated by using the Structural Similarity Index (SSIM) (4) which measures the similarity between the reference image (i.e the last cycle) and the registered cycles by considering the texture information. This computation results in a value between 0-1 where 0 implies a completely different texture, 1 is achieved when there is a perfect match. In the QC plots, we show the scores for every cycle before and after registration where we choose 0.5 as an ad-hoc threshold to evaluate the registration quality of a run.

In-house spatial transcriptomics method protocol

Samples S1-S4 were generated with an in-house spatial transcriptomics method that is based on Slide-Seq and Slide-SeqV2 protocols: an array with beads arranged on it is first generated, termed puck. The cellular barcodes are then optically decoded with a microscope as described in Slide-SeqV2 (5). ERCC RNA Spike-In Mix and a GFP reporter for S1 or E12 mouse brain sections for S2-S4 were subsequently placed on the optically decoded pucks and spatial transcriptomics libraries were generated following (5, 6). The prepared libraries were finally sequenced with NextSeq 500 Illumina machines and analysed as described below.

Obtaining the Slide-Seq & Slide-SeqV2 optical sequencing barcodes

For Slide-Seq samples, we have downloaded the processed Puckcaller files from (https://singlecell.broadinstitute.org/single_cell/study/SCP354/slide-seq-study) and extracted the bead barcodes listed in the published AllBeadBarcodes.csv files for each sample. If there were multiple runs, we used the one with the latest date stamp. For the Slide-SeqV2 sample, we used the corresponding bead locations file from (https://singlecell.broadinstitute.org/single_cell/study/SCP815/highly-sensitive-spatial-transcriptomics-at-near-cellular-resolution-with-slide-seqv2).

Processing & analysis of spatial transcriptomics sequencing data

To obtain the set of true barcodes of the samples S1-S4 we processed and analyzed the raw fastq files with spacemake v.0.4.2 with default parameters (7). The cell barcode was extracted from read1 as the reverse sequence of nucleotide positions 9-20, while the UMI was extracted from read1 from nucleotide positions 1-8. Read2 was aligned to a combined genome of ERCC RNA Spike-In Mix and GFP reporter for S1 and to the mm10 genome for S2-S4. To obtain the set of true barcodes of the Slide-Seq and Slide-SeqV2 samples we used the tool DigitalGeneExpression from Drop-seq tools v.2.5.0 with default parameters to generate the digital gene expression matrix for the top 100,000 cell barcodes.
While the extracted top 100,000 cell barcodes are used for all the processing in the pipeline, we also generated the top 300,000 barcodes to analyse the effect of the number of top Illumina barcodes on the barcode matching process (Sup. Fig. 8). For the samples S1-S4, spacemake output as explained above was used. For Slide-Seq and Slide-SeqV2 samples, we used the DigitalGeneExpression tool as explained above, but with the barcode cutoff 300,000. It should be noted that due to the diverse read mappability, the top 100,000 of this new set does not perfectly overlap with the original set of barcodes.

Comparing Illumina-sequenced and the optically sequenced barcodes

After the Digital Gene Expression matrices were generated, we selected the top 100,000 bead barcodes with the highest number of counts to compare their nucleotide sequences against the sequences obtained by processing the optical sequencing data. For samples S1-S4 and Slide-SeqV2 samples, we counted exact matches between the two sets of barcodes. For Slide-Seq samples, images are generated through SOLiD sequencing and therefore they do not directly map to the nucleotides outputted by Optocoder. We have included a custom script that can convert the Illumina barcodes to the colour space by using the correct image and ligation sequences. Essentially, we ran Optocoder on all 20 images and took 14 non-constant cycles that are part of the barcode. While Optocoder outputs barcodes with nucleotides, we converted these barcodes to the numerical indices to represent them in the colour space. Then, the Illumina barcodes are converted to the colour space using the ligation sequences and orders as defined in https://github.com/MacoskoLab/PuckCaller and (6). Finally, we counted exact matches between the two sets in colour space.

Machine Learning Basecaller

Dataset

Machine learning basecaller is trained for every dataset separately provided that the respective Illumina sequencing run is available. Optocoder first calculates the matching barcodes between the optically decoded barcodes (after phasing correction) and the top 100,000 barcodes from the sequencing data. Specifically, input features for a sample is $Beads^{Matching} = \{(B^s_i, bc_i) \mid \forall bc_i \in SB\}$ where B^s_i is intensity matrix of bead *i* after robust scaling, bc_i is the barcode corresponding to bead *i*, and SB is the set of top 100,000 Illumina sequenced barcodes. Input dataset $Beads^{Matching}$ is used for training and is split into training (80%) and validation (20%) sets, and the non-matching barcodes are used as the test set.

Models and Training

Optocoder implements four different types of classifiers. Random Forest (RF) and Multilayer Perceptron (MLP) models are implemented using scikit-learn (v0.22.2.post1), Gradient Boosting (GB) is implemented using XGBoost (v1.1.1), and finally a Recurrent Neural Network (RNN) model is implemented using Keras (v2.3.1) and Tensorflow (v2.1.0).
For RF, MLP and GB models, Optocoder uses scikit-learn's Multi Output Classifier approach where a classifier for each target is fitted. Here, one classifier for each sequencing cycle was trained. A random search was done with 10 iterations to optimise hyperparameters. Parameter ranges for the search are defined in the Optocoder machine learning module and the parameters of the selected models for the data presented in this paper are listed in Sup. Table 1. For the RNN model, we have one bidirectional LSTM layer, followed by dropout and a dense layer with softmax activation function. The model is trained using Adam optimizer and sparse categorical entropy loss. For every fit, 30 epochs with batch size 50 is used along with an early stopping mechanism. All samples were trained with these parameters. Furthermore, KerasTuner (v1.0.1) is used for the random hyperparameter search with 10 iterations. After training the models, new barcodes are predicted for non-matching beads. Optocoder saves the barcodes for all four models on disk, but the main basecaller is Gradient Boosting, as it consistently outperforms the others. Optionally, Optocoder machine learning module can be run with k-fold cross validation for parameter search and training steps.

Barcode Prediction and Matching

After training the models, new barcode sequences are predicted for all optically decoded bead barcodes that do not match to any of the top 100,000 Illumina sequenced barcodes.

The predicted barcodes that match to a sequenced barcode are then retained and added to the set of already matching barcodes.

Specificity Analysis of Machine Learning Models

To analyse the number of mismatches in the training set, we first predicted the barcodes in the training set using the trained models for the respective puck. Then, we calculated the difference between the original number of matching barcodes in the training set and the number of matches after the predictions.

Score Thresholding Analysis

For the analysis of the chastity score thresholds, we calculated the average score for every barcode in each method. Then, barcodes were filtered for the scores starting from 0.5, which is the lower bound of the chastity score, to 1.0 with 0.05 increments. Remaining barcodes after filtering were matched to the top 100,000 illumina barcodes and number of matches were calculated. For the machine learning models, we followed the same procedure but instead of the chastity score, we used the prediction score of the model for the most probable bases.

Bibliography

1. Evangelidis,G.D. and Psarakis,E.Z. (2008) Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Trans. Pattern Anal. Mach. Intell.*, **30**, 1858–1865.
2. Li,L. and Speed,T.P. (1999) An estimate of the crosstalk matrix in four-dye fluorescence-based DNA sequencing. *Electrophoresis*, **20**, 1433–1442.
3. Stanford University,D.T.,Govinda Kamath, Jesse Zhang Lecture 3: Base Calling for Second-Generation Sequencing. http://data-science-sequencing.github.io/Win2018/lectures/lecture3/ Last accessed: 28.04.2022
4. Wang,Z., Bovik,A.C., Sheikh,H.R. and Simoncelli,E.P. (2004) Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, **13**, 600–612.
5. Stickels,R.R., Murray,E., Kumar,P., Li,J., Marshall,J.L., Di Bella,D.J., Arlotta,P., Macosko,E.Z. and Chen,F. (2021) Highly sensitive spatial transcriptomics at near-cellular resolution with Slide-seqV2. *Nat. Biotechnol.*, **39**, 313–319.
6. Rodriques,S.G., Stickels,R.R., Goeva,A., Martin,C.A., Murray,E., Vanderburg,C.R., Welch,J., Chen,L.M., Chen,F. and Macosko,E.Z. (2019) Slide-seq: A scalable technology for measuring genome-wide expression at high spatial resolution. *Science*, **363**, 1463–1467.
7. Sztanka-Toth,T.R., Jens,M., Karaiskos,N. and Rajewsky,N. (2021) Spacemake: processing and analysis of large-scale spatial transcriptomics data. *bioRxiv*.

Sup. Table 1. Machine Learning Model Parameters for the trained models

	P1	**P2**	**P3**	**P4**
XGB Classifier	**colsample_bytree** 0.795, **learning_rate** 0.236, **max_depth** 3, **min_child_weight** 4, **n_estimators** 911, **subsample** 0.849, **tree_method** 'hist'	**colsample_bytree** 0.767, **learning_rate** 0.517, **max_depth** 5, **min_child_weight** 3, **n_estimators** 477, **subsample** 0.781, **tree_method** 'hist'	**colsample_bytree** 0.817, **learning_rate** 0.070, **max_depth** 4, **min_child_weight** 4, **n_estimators** 489, **subsample** 0.396, **tree_method** 'hist'	**colsample_bytree** 0.855, **learning_rate** 0.213, **max_depth** 6, **min_child_weight** 1, **n_estimators** 550, **subsample** 0.734, **tree_method** 'hist'
Random Forest Classifier	**bootstrap** False, **max_depth** 110, **max_features** 'auto', **min_samples_leaf** 1, **min_samples_split** 2, **n_estimators** 180	**bootstrap** False, **max_depth** 70, **max_features** 'auto', **min_samples_leaf** 2, **min_samples_split** 2, **n_estimators** 500	**bootstrap** False, **max_depth** 50, **max_features** 'auto', **min_samples_leaf** 1, **min_samples_split** 5, **n_estimators** 180	**bootstrap** False, **max_depth** 90, **max_features** 'sqrt', **min_samples_leaf** 1, **min_samples_split** 10, **n_estimators** 1000
MLP Classifier	**hidden_layer_sizes** (100,), **activation** 'tanh', **solver** 'adam', **alpha** 0.05, **learning_rate** 'constant', **max_iter** 200	**hidden_layer_sizes** (100,), **activation** 'tanh', **solver** 'adam', **alpha** 0.1, **learning_rate** 'adaptive', **max_iter** 200	**hidden_layer_sizes** (50, 100, 50), **activation** 'tanh', **solver** 'sgd', **alpha** 0.1, **learning_rate** 'adaptive', **max_iter** 1000	**hidden_layer_sizes** (100,), **activation** 'relu', **solver** 'adam', **alpha** 0.05, **learning_rate** 'constant', **max_iter** 1000
RNN Classifier	**units**: 64 **dropout:** 0.1 **learning_rate:** 0.01	**units:** 288 **dropout:** 0.2 **learning_rate:** 0.01	**units:** 448 **dropout:** 0.3 **learning_rate:** 0.001	**units:** 256 **dropout:** 0.2 **learning_rate:** 0.01

	SSP1	**SSP2**	**SSP3**	**SSP4**
XGBClassifier	**colsample_bytree** 0.639, **learning_rate** 0.410, **max_depth** 3, **min_child_weight** 1, **n_estimators** 742, **subsample** 0.732, **tree_method** 'hist'	**colsample_bytree** 0.822, **learning_rate** 0.278, **max_depth** 5, **min_child_weight** 1, **n_estimators** 739, **subsample** 0.726, **tree_method** 'hist'	**colsample_bytree** 0.832, **learning_rate** 0.316, **max_depth** 4, **min_child_weight** 4, **n_estimators** 735, **subsample** 0.825, **tree_method** 'hist'	**colsample_bytree** 0.593, **learning_rate** 0.366, **max_depth** 3, **min_child_weight** 2, **n_estimators** 754, **subsample** 0.772, **tree_method** 'hist'
Random ForestClassifier	**bootstrap** False, **max_depth** 70, **max_features** 'sqrt', **min_samples_leaf** 2, **min_samples_split** 2, **n_estimators** 1000	**bootstrap** False, **max_depth** 60, **max_features** 'sqrt', **min_samples_leaf** 1, **min_samples_split** 5, **n_estimators** 130	**bootstrap** False, **max_depth** 100, **max_features** 'sqrt', **min_samples_leaf** 1, **min_samples_split** 2, **n_estimators** 500	**bootstrap** False, **max_depth** 60, **max_features** 'sqrt', **min_samples_leaf** 1, **min_samples_split** 2, **n_estimators** 500
MLPClassifier	**hidden_layer_sizes** (50, 100, 50), **activation** 'tanh', **solver** 'sgd', **alpha** 0.1, **learning_rate** 'constant', **max_iter** 600	**hidden_layer_sizes** (100,), **activation** 'tanh', **solver** 'adam', **alpha** 0.0001, **learning_rate** 'adaptive', **max_iter** 600	**hidden_layer_sizes** (50, 100, 50), **activation** tanh', **solver** 'sgd', **alpha** 0.05, **learning_rate** 'adaptive', **max_iter** 400	**hidden_layer_sizes** (100,), **activation** 'tanh', **solver** ‘adam', **alpha** 0.0001, **learning_rate** ‘constant', **max_iter** 200
RNN Classifier	**units:** 288 **dropout:** 0.3 **learning_rate:** 0.001	**units:** 448 **dropout:** 0.3 **learning_rate**: 0.01	**units:** 448 **dropout:** 0.1 **learning_rate:** 0.01	**units:** 352 **dropout:** 0.4 **learning_rate:** 0.001

Sup. Table 2. Slide-Seq and Slide-Seq Data

Puck Name (Optocoder)	**Puck ID**	**Slide-Seq Version**
SSP1	180413_7	V1 (SOLiD)
SSP2	180430_1	V1 (SOLiD)
SSP3	180528_23	V1 (SOLiD)
SSP4	200115_08	V2