Supplementary Methods

Machine Learning Models and CNN

Unsupervised Machine Learning was carried out using PCA and k-means clustering algorithms. PCA was used to reduce dimensionality[32] to 36 principal components (PCs) that explained 95.1% of the variance in the data. K- means was carried out varying k from 2 to 20 on the normalized values of the dimension-reduced tiles. Each cluster was classified as rotational or non-rotational based on majority designation.

The first supervised machine learning model we used was LDA. In order to increase the training size, the model was trained on the training and validation cohorts since there are no hyper-parameters to tune in this network. Once developed, the model was evaluated on the testing set.

The second supervised machine learning approach used was k-NN. Analogous to k-means, we used the absolute values of the dimension-reduced tiles as the input to the k-NN algorithm as described previously. The training set was used for training the model and the validation set was used to optimize the value of the hyper-parameter k. The model was run with increasing numbers of k until the error in the validation set converged. The value of k was then chosen where the validation set had the least error. The final model was then run with the chosen value of k on all training, validation, and testing groups.

The third supervised machine learning approach we used was SVM. Similar to K-NN and LDA, we used the absolute values of the dimension-reduced tiles as the input to the SVM algorithm. We used the training set to train the SVM model with a linear kernel and a

regularization factor (C=1). The trained model was then run on all training, validation, and testing groups.

The fourth supervised model was implemented using a CNN. AlexNet [45] was chosen as the base neural network design, with minor modifications to accommodate the dimensions of our inputs. The network consists of 5 convolutional layers and 3 fully connected layers, an input layer, an output layer, as well as multiple activation, max-pooling, and batch normalization layers. The input to the network was a single 13x13x3 RGB tile as given by the phase mapping software. The network's final layer was a binary cross-entropy loss function to output either rotational (1) or non-rotational (0). The detailed architecture of the CNN is shown in figure 2. The CNN weights were updated through backpropagation of the loss function, using mini-batch stochastic gradient descent with learning rate = 0.01 and momentum = 0.9. During training, we used early stopping on the validation set to ensure that the CNN did not overfit to the training set.