Problems with k-nearest neighbors is that predicting new instances of data sample is slow because it requires us to compare the sample with all the images in the training set. The classifier must store all the of the training data which is unrealistic because datasets can easily go up to terabytes in size. A better model for image classification is a linear classification model

We define a linear classification model as $f(x_i, W\, b) = Wx_i + b$ . The function takes in a weight vector W that we have learned, a bias vector b which influences the output score but doesn't interact with the data and the specific image we want to predict $x_i$. The advantage of this model as opposed to the k-nearest neighbors is that once we obtained the optimal $W, b$ from training we no longer need to carry the training data around. Another improvement of the linear classification model is that to predict the new image $x_i$ class we only need to compute the matrix operations $Wx_i + b$ which is must faster than comparing the test image to all images in the training set.

In machine learning it is a common practice to perform normalization on your design matrix $W$ so that it performs better when applying techniques like gradient descent. One of the most common form of data preprocessing is zero mean centering.

We don't have control over the training data that we recieve but we do have control over the parameters $W$, $b$ that we learn. We can define a metric to see how well our current parameters $W$, $b$ perform on our dataset as a loss function. A loss function would be high if we perform poorly with the training data using our current parameters and should be low if we perform well. Different machine learning models have different loss functions. The process of determining parameters that lead to optimals $W$, $b$ given our loss function is call called optimization.