6.867 - Homework 2

Anonymous

I. Interpretation of Ridge Regression

1) a) In the first part of this problem, we center the data and the target, resulting in new matrices C and Z respectively. We then augment the centered matrix C with d additional rows $\sqrt{\lambda}I$ and augment Z with d additional rows of zeros.
From there, we minimize the loss function with our new data and target, replacing the standard data and target as shown:

$$min_w L(w) = \frac{1}{N}\sum(z^{(i)} - w^T\phi(c^{(i)})^2 + \lambda||w||^2$$

From here we take the gradient of the loss function and solve for w:
$\bigtriangledown L(w) = 2C^TCw - 2C^Tz + 2\lambda w = 0$

$2C^TCw - 2C^Tz + 2\lambda w = 0 \rightarrow 2C^TCw + 2\lambda w = 2C^Tz$

$2C^TCw + 2\lambda w = 2C^Tz \rightarrow w* = (C^TC + \lambda I)^{-1}C^Tz$

Proving that the closed-form solution is $w* = (C^TC + \lambda I)^{-1}C^Tz$

b) We are given that the prior distribution is:

$w \sim N(0, \tau^2 I)$ and that the likelihood is:

$y \sim N(Xw, \sigma^2 I)$

We know that the posterior is correlated to the likelihood times the prior distributions, or:

$$P(Posterior) \propto P(Likelihood)P(Prior)$$

We use this to solve for the posterior distribution using the known equations of the normal distribution.

$$P(Posterior) \propto P(Likelihood)P(Prior) = e^{\frac{-1}{2\sigma^2}(y-Xw)^T(y-Xw)}e^{\frac{-w^Tw}{2\tau^2}} = e^{\frac{-1}{2\sigma^2}(y-Xw)^T(y-Xw) - \frac{w^Tw}{2\tau^2}}$$

To simplify, we take the log of both sides:

$$logP(Posterior) \propto \frac{-1}{2\sigma^2}(y-Xw)^T(y-Xw) - \frac{w^Tw}{2\tau^2}$$

We then take the derivative of the $logP(Posterior)$ to solve for w:

$\frac{dlogP(Posterior)}{dw} = \frac{1}{\sigma^2}X^T(y - Xw) - \frac{w}{\tau^2} = 0$

$\frac{1}{\sigma^2}X^T(y - Xw) = \frac{w}{\tau^2} \rightarrow X^T(y - Xw) = \frac{\sigma^2 w}{\tau^2}$

$X^T(y - Xw) = \frac{\sigma^2 w}{\tau^2} \rightarrow X^Ty - X^TXw = \frac{\sigma^2 w}{\tau^2}$

$X^Ty = w(X^TX + \frac{\sigma^2}{\tau^2}) \rightarrow w* = (X^TX + \frac{\sigma^2}{\tau^2})^{-1}X^Ty$

Here, we've proven that the closed-form solution of ridge regression can be derived from Bayesian regression, with $\lambda = \frac{\sigma^2}{\tau^2}$. From here, we can also show that $w*$ is the mean of the posterior distribution. We assume that the mean of the posterior distribution is m_N and the variance is S_N, and write the exponent of the normal distribution as:

$\frac{1}{2}(w - m_N)^T S_N^{-1}(w - m_N) = \frac{1}{2}(w^T S_N w - 2w^T S_N^{-1} m_N + m_N^T S_N^{-1} m_n$

We can find the inverse of the variance with the first term on the right hand side:

$w^T S_N w = \frac{1}{\sigma^2}w^T X^T X w + \frac{1}{\tau^2}w^T w = \frac{1}{\sigma^2}w^T(X^TX + \frac{\sigma^2}{\tau^2}I)w \rightarrow S_N^{-1} = \frac{1}{\sigma^2}(X^TX + \frac{\sigma^2}{\tau^2}I)$

We then find the mean by plugging w back in as the mean:

$m_N = (X^TX + \frac{\sigma^2}{\tau^2})^{-1}X^Ty = \frac{1}{\sigma^2}S_N X^T y$

$w^T S_N w \rightarrow w^T S_N m_N = \frac{1}{\sigma^2}w^T S_N^{-1} S_N X^T y = \frac{1}{\sigma^2}w^T N X^T y = w^T S_N w$

Therefore, the closed form solution is also the mean of the posterior distribution. By definition, the mean of a Gaussian is also the mode.

II. Neural Networks with One Hidden Layer Can Approximate Continuous Function

1) By definition, any function in $span(H_\rho)$ is less than some distance ϵ away from another function in $span(H_\rho)$. This also applies to functions in $span(H_\phi$. That is:

for $h(x)$ and $f(x) \in span(H_\rho)$:

$sup(|h(x) - f(x)| : x \in [r, s]) \leq \tau$

and for $g(x)$ and $\phi(x) \in span(H_\rho)$:

$sup(|g(x) - \phi(x)| : x \in [r, s]) \leq \tau$

We are given that $sup(|h(x) - \phi(x)| : x \in [r, s]) \leq \tau$, so based on the above known relationships, it follows that:

$sup(|f(x) - g(x)| : x \in [r, s]) \leq \tau$

Given that the above is defined as applying to any interval $[r, s]$, it also follows that $[r, s] = [0, 1]^d$ because of the stipulation of "any interval", proving that:

$sup(|f(x) - g(x)| : x \in [0, 1]^d) \leq \epsilon$

where ϵ is also some value greater than zero.

2) We can prove that the exponential function is Lipschitz continuous on any bounded interval:

$|e^{x_2} - e^{x_1}| \leq K|x_2 - x_1|$

by rearranging the relationship:

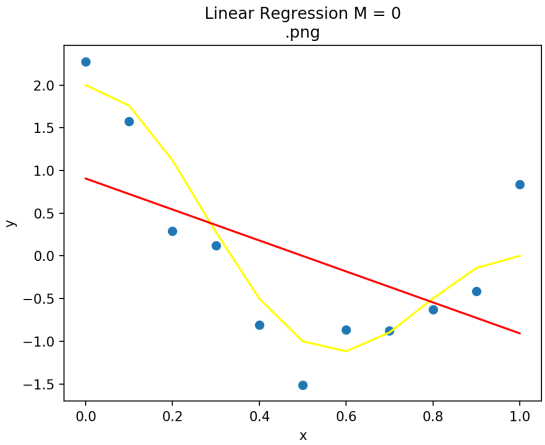
$log|e^{x_2} - e^{x_1}| \leq log(K) + log(|x_2 - x_1|)$

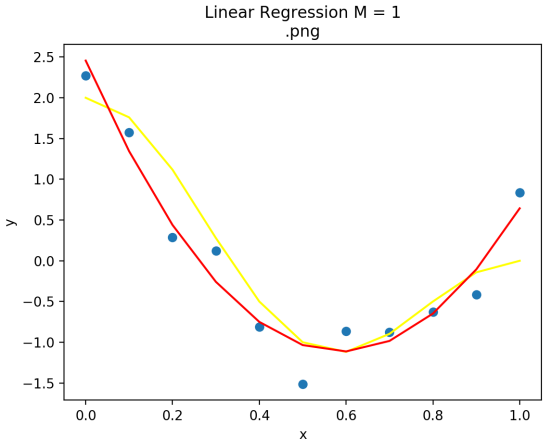
$log|e^{x_2} - e^{x_1}| - log(|x_2 - x_1|) \leq log(K)$

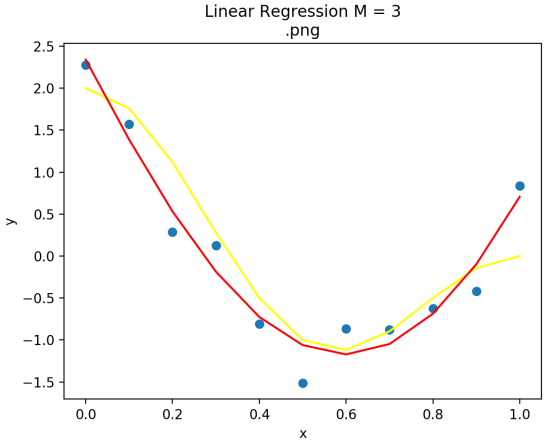
Assuming that this function is bounded, $log|e^{x_2} - e^{x_1}|$ is larger than $log(|x_2 - x_1|)$. In addition, because they are bounded by absolute values before the logarithm is applied, they will also be real-valued. Based on this, $log(K)$ both positive (because it must be for the log function to apply) and real-valued (because the difference must be real-valued).

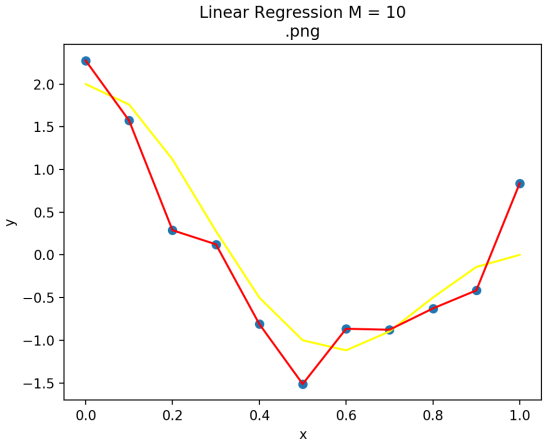
III. Linear Basis Function Regression

1) Replicates of the plots in figure 1 are shown below, with the optimal weight vector in red, the data in blue, and true function in yellow:



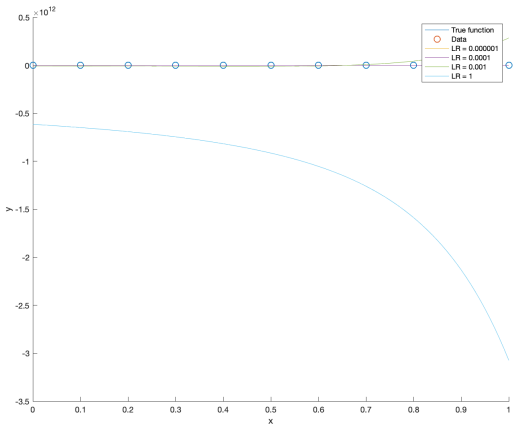




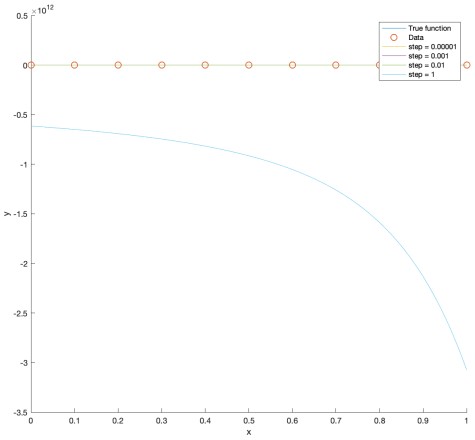


2) Comparing the batch gradient descent and the stochastic gradient descent across varying hyperparameters (all with polynomial order 10), we can see that both optimizers perform similarly in the case of high step size (for batch gradient descent) and high learning rate (stochastic gradient descent). This is not surprising, as the learning rate and step size serve similar roles in optimization (as we'll see in Alice_Net), and both too high and too low values for either can result in poor model accuracy/fit.

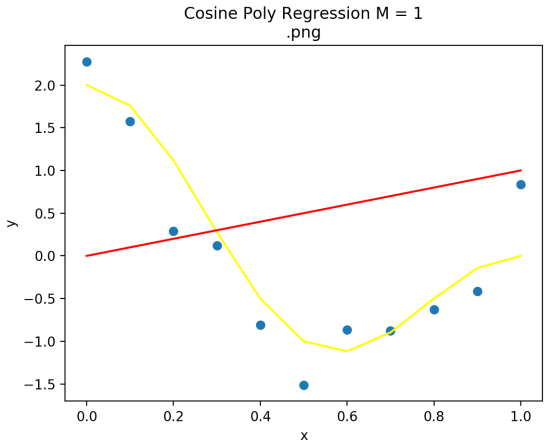
SSE with Stochastic Gradient Descent and Varied Learning Rate

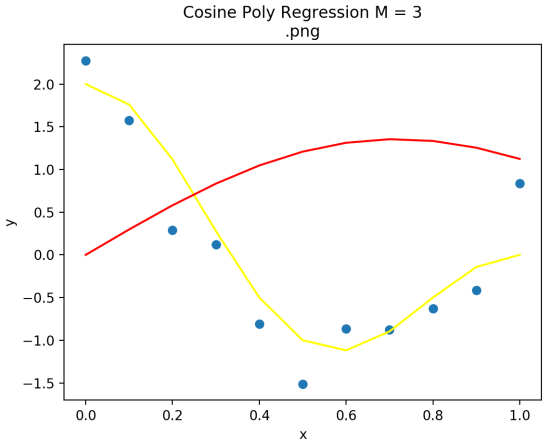


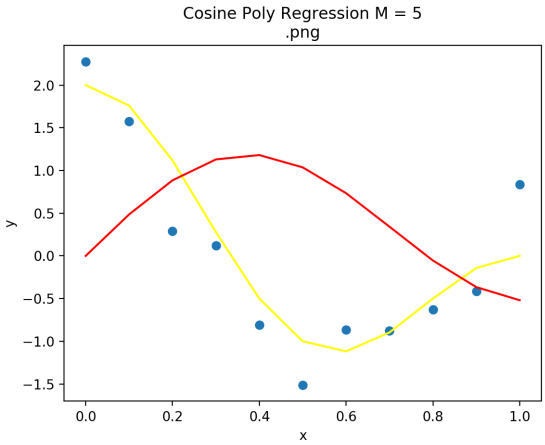
SSE with Batch Gradient Descent and Varied Step Size

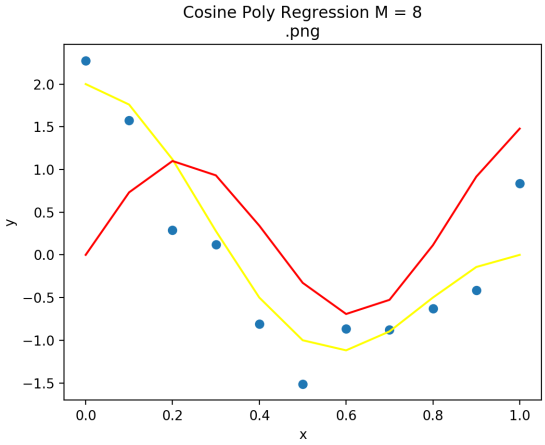


3) Plots of the optimal weight vector with cosine basis functions are shown below with the same color scheme as the plots in part a:





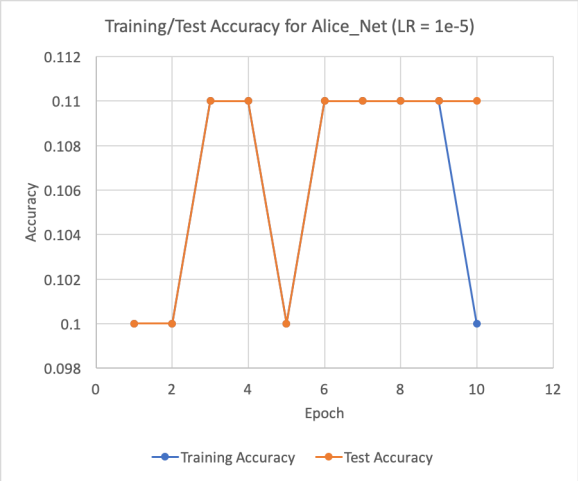




These weight vectors come to the true function (especially at higher values of M), but still do not accurately reproduce the true function that was used to create the data.

IV. Neural Network - What Went Wrong?

1) a) Graphs of the training and test error for Alice_Net are shown below:

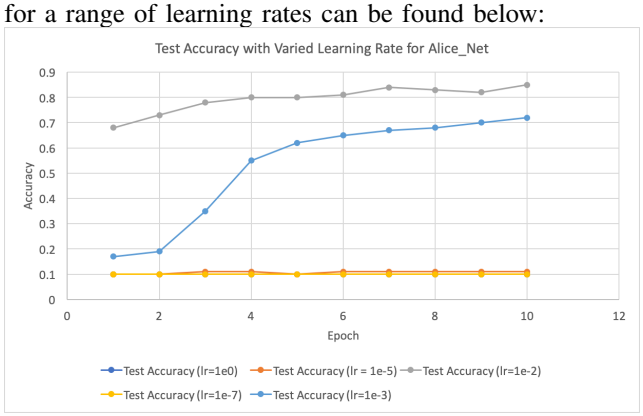


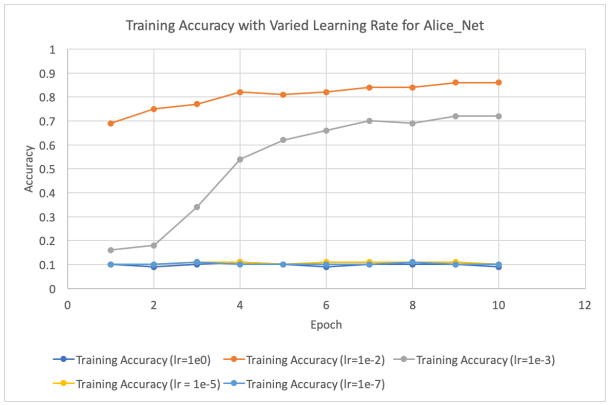
Alice's training accuracy and test accuracy after 10 epochs were 0.1 (10% accuracy, 90% error) and 0.11 (11% accuracy, 89% error) respectively.

Alice's model had an input layer, two convolutional layers, and two fully connected layers that led to 10 probabilities as an output, where the class with the highest probability was chosen by the model. Her hyperparameters were: a kernel size of 5 for all convolutional layers, the first convolutional filter layer was 1x10, the second convolutional layer filter was 10x20, the first fully connected layer was 320 to 50, and the second fully connected layer was 50 to 10. Her stride length was 2, her learning rate was 0.00001, and her number of epochs was 10.

b) Alice set the learning rate of her model to be too small of a value, which was prohibiting the model from making significant progress in 10 epochs. I fixed this problem by increasing the learning rate by a factor of 1000, resulting in a test accuracy of 0.85 (error of 15%). Graphs of the rates used can be found in part C.

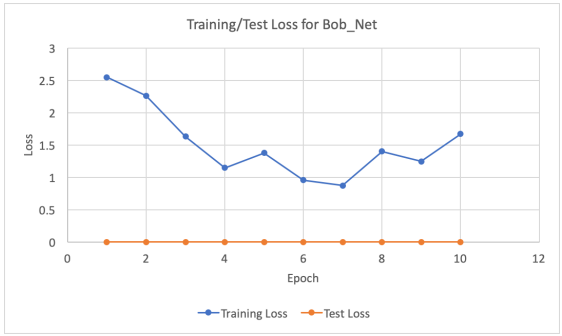
c) Plots of the training and test losses and accuracies for a range of learning rates can be found below:

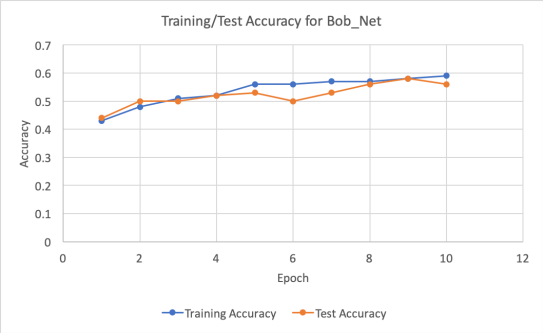




In the case of values being too high or too low, the accuracy of the model remained at around 10% after 10 epochs, and the loss remained high. This shows that optimization of the learning rate is extremely important, as both too high and too low of a learning rate results in poor accuracy and high loss.

2) a) Below are the graphs of accuracy and loss for Bob_Net:





The training accuracy was 0.59 (59%) after 10 epochs, and the test error was 0.56 (56%).

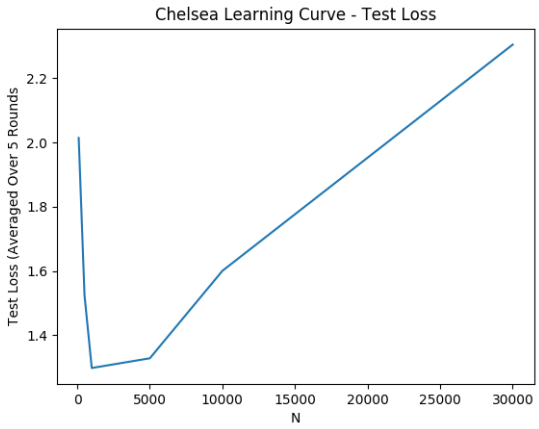
Bob's model had an input layer, two convolutional layers (1x10 with kernel size of 5 and 10x20 with kernel size of 5), and two fully connected layers (320 to 50 neurons and 50 to 1 neuron) that output a single number. He chose 10 epochs and a learning rate of 0.01 as his hyperparameters.

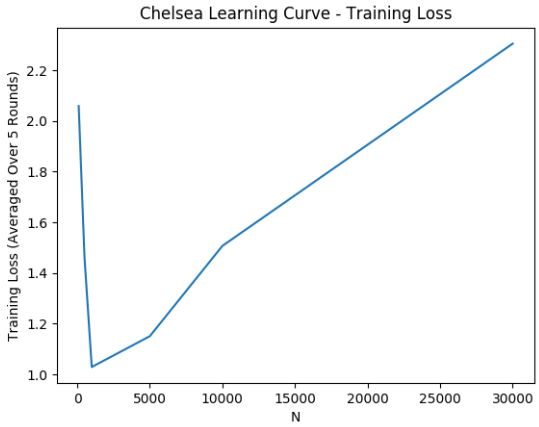
b) Bob is trying to make a binary classifier, shown in the test and train methods when he takes the single probability output of the final fully connected layer and rounds either up or down (pred = torch.round(output), seen on line 34 in train() and line 60 in test()) based on the model's predictions. He also uses Mean Squared Error Loss instead of Negative Log Likelihood, which may cause his model to converge slower (lines 33 and 59, loss = F.mse_loss(output.view(target.shape), target.float())).

c) Bob was attempting to make a binary classifier, which can be fixed by changing his last fully connected letter to outputting 10 classes instead of one, as well as by changing his accuracy calculation (shown in part b) to find the index with the highest probability instead of rounding the single output probability to determine whether the image fits into one of two categories. This fix would work better because it would allow Bob to accurately classify the images into the correct categories. He should also switch his loss function to Negative Log Likelihood to speed up the time it takes for his model to converge.

3) a) I do not believe her, as such a small number of training samples would likely overfit the model to the training samples (especially in a multi-class classifier with 10 classes), so any positive results that she is seeing are likely due to overfitting on that set. While it is possible that less than 60,000 samples are needed to adequately train this model, it is very likely that the model needs more than 100 samples.

b) The learning curves are shown below, where the losses are averaged over five iterations of the model (1 iteration = 10 epochs):





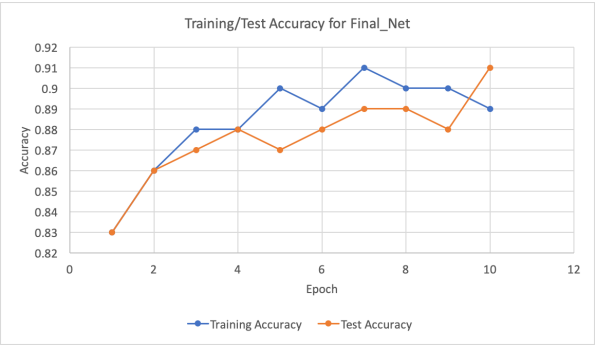
Based on these learning curves, the model needs roughly N=2500 samples to be adequately trained in it's current state.

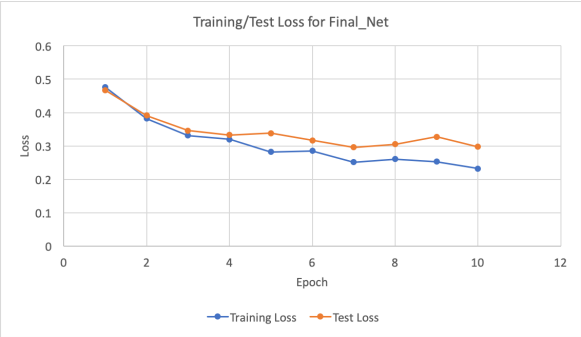
4) a) The final training accuracy of Daniel's model was 0.89 (89%).

The final test accuracy of Daniel's model was 0.91 (91%).

The final accuracy on Daniel's photos was 0.50 (50%).

The final accuracy and loss for the training and test data is shown below:





b) In looking at the training and test data, very few of the images have been rotated or perturbed in any way. However, Daniel's data set is largely comprised of rotated and perturbed images. This is likely the cause of the low accuracy of his model on real-world data, as the model has not learned to identify rotated or perturbed clothing. This problem can be solved by perturbing the training and test data using image transformations, which both expands the available dataset and provides examples that are more similar to real-world images.