Question 3:

For this question, I choose ClippedGradientDescentOptimizer and tanh for all models.

(c)
For the first RNN model (with one hidden layer):
Sparse_softmax_cross_entropy_with_logits loss function does really nice job in RNN model with one hidden layer. It gets smallest misclassification errors on training and validation data when compared to sparse_kl_divergence_rl and sparse_kl_divergence_ml. Sparse_kl_divergence_rl loss function gets largerst misclassification errors on training and validation data. The model gets smaller misclassification errors on training data also gets smaller misclassification errors on validation error, so the training and validation performance are monotonically related.

(e)
For the LSTM model:
Sparse_softmax_cross_entropy_with_logits loss function gets the best result in validation and test data (lower misclassification errors on training and validation data). The performance of sparse_kl_divergence_ml loss function is just a little bit worse than Sparse_softmax_cross_entropy_with_logits loss function. Sparse_kl_divergence_rl loss function gets largerst misclassification errors on training and validation data. The model gets smaller misclassification errors on training data also gets smaller misclassification errors on validation error, so the training and validation performance are monotonically related.

(g)
For the second RNN model (with two hidden layers)
Sparse_softmax_cross_entropy_with_logits loss function does really nice job in RNN model with one hidden layer. It gets smallest misclassification errors on training and validation data when compared to sparse_kl_divergence_rl and sparse_kl_divergence_ml. Sparse_kl_divergence_rl loss function gets largerst misclassification errors on training and validation data. The model gets smaller misclassification errors on training data also gets

smaller misclassification errors on validation error, so the training and validation performance are monotonically related.

(h)
The final results of three models with three different loss functions are as follows:

model	step	train_loss	train_err	valid_loss	valid_err	epoch_time
LSTM_base	127540	1.3027	0.283546	1.4227	0.304379	7480.69
RNN2_base	127540	1.38568	0.298178	1.51345	0.32019	9513.66
LSTM_kl_ml	127540	0.0994748	0.314044	0.104621	0.325681	7677.28
RNN1_base	127540	1.56481	0.329701	1.64683	0.344392	4084.42
LSTM_kl_rl	127540	0.0226817	0.387512	0.0227771	0.389887	7668.64
RNN2_kl_ml	127540	0.181735	0.404065	0.183018	0.406066	9722.94
RNN1_kl_ml	127540	0.184872	0.557938	0.184495	0.556652	4280.22
RNN2_kl_rl	127540	0.0325836	0.626907	0.0325404	0.625608	9743.9
RNN1_kl_rl	127540	0.032878	0.636652	0.0328265	0.635453	4293.41

When compare these three models, I found no matter which model I choose, Sparse_softmax_cross_entropy_with_logits always gets the best result, sparse_kl_divergence_ml is the second,
and sparse_kl_divergence_rl is the worst choice. When compare different models with same loss function, I found that LSTM always gets the best result, RNN with 2 hidden layers is the second, RNN with 1 hidden layer is the worst choice. In all, LSTM
with Sparse_softmax_cross_entropy_with_logits loss function gets smallest misclassification errors on training and validation data, while RNN (1 hidden layer) with sparse_kl_divergence_rl loss function gets largest misclassification errors on training and validation data. The model gets smaller misclassification errors on training data also gets smaller misclassification errors on validation error, so the training and validation performance are monotonically related.

(i)
Misclassification rate only cares about whether the predicted character is right or not,
while Sparse_softmax_cross_entropy_with_logits loss function computes sparse softmax cross entropy between logits and labels, it means this loss function considers the probability of the predicted

character. The advantage of using misclassification rate is that it can show the result directly, because what we really care about is just whether the predicted character is right or not, however, using misclassification rate also loss some information. Sometimes though the predicted character is not right, the right character is the just the second choice of the model, we may can also consider it's a good prediction. Misclassification rate can not capture this kind of information, while Sparse_softmax_cross_entropy_with_logits loss function is able to capture it.