Question 4:

(c)
For normal MNIST images data, convolutional models get better results (lower error) in validation and test data compared to fully connected models. However, the train loss of convolutional models are significantly larger than their corresponding fully connected models.

For MNIST images where pixels have been permuted, fully connected model gets better results (lower error) in all three aspects (train loss, validation error and test error.) compared to convolutional model.

Besides, the result gained by fully connected model on permuted images data is only a little bit worse than that gained on normal images data. However, the result gained by convolutional model on permuted images is much worse than that gained on normal images data.
This result shows that convolutional model can't generalize well when data is permuted.

The reason why convolutional model behave so differently on two data sets: Each templet of convolutional model only focus on small parts of image, such as 3*3 and 5*5. If there exists some noisy data in such a small area, it will have large impact on the representation of this area, and the representation of this area can not represent this area's true property, it might result in overfitting problem finally.

For fully connected model, the reason why its behavior doesn't change much is because fully connected model focus on the whole area of the image, some noisy in the image can not have much impact on the representation of the image, so the final result does not change much.

(d)
From my perspective, the reason why convolutional model behave not very well is because it overfits data, so I want to check the L2 loss of the model. It's very easy to implement, the code I write is as follows:

```
self.weight_loss = tf.constant(0.0)
self.weight_loss = self.weight_loss + tf.nn.l2_loss(W_1) + tf.nn.l2_loss(W_2) + tf.nn.l2_loss(W_3)
self.weight_loss = self.weight_loss + tf.nn.l2_loss(b_1) + tf.nn.l2_loss(b_2) + tf.nn.l2_loss(b_3)
```

It's very efficient, we can get this value when we get train loss value.

(e)
After implement this method, I think the result shows this method can predict successful generalization more or less.

For fully connected model and normal data, at the beginning of the train period, the L2 loss is 285089, after train finished, the L2 loss is 336613, the L2 loss increased by about 18%.

For convolutional model and normal data, at the beginning of the train period, the L2 loss is 37844.7, after train finished, the L2 loss is 42899.3, the L2 loss increased by about 13%.

For fully connected model and permuted data, at the beginning of the train period, the L2 loss is 284580, after train finished, the L2 loss is 331643, the L2 loss increased by about 16.5%.

For convolutional model and normal data, at the beginning of the train period, the L2 loss is 37265.7, after train finished, the L2 loss is 49913.8, the L2 loss increased by about 34%.

The generalization ranking should be 1. convolutional model and normal data 2. fully connected model and normal data 3. fully connected model and permuted data 4. convolutional model and normal data

The increase of L2 loss can show the overfitting problem more or less, the increase of L2 loss ranking is 1. convolutional model and normal data 2. fully connected model and permuted data 3. fully connected model and normal data 4. convolutional model and normal data

The only one exceptional case is that fully connected model and normal data should generalize better than fully connected model and permuted data, but the L2 loss ranking predicts fully connected model and permuted data generalize better than fully connected model and normal data, though the difference is really small. Therefore, I think my method can predict successful generalization more or less.