


Migration of model architectures

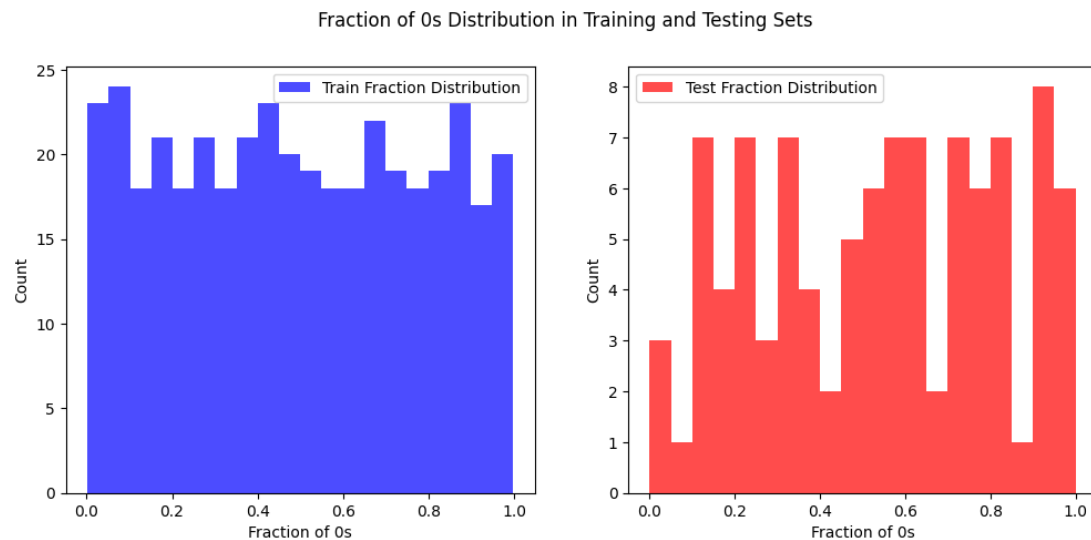
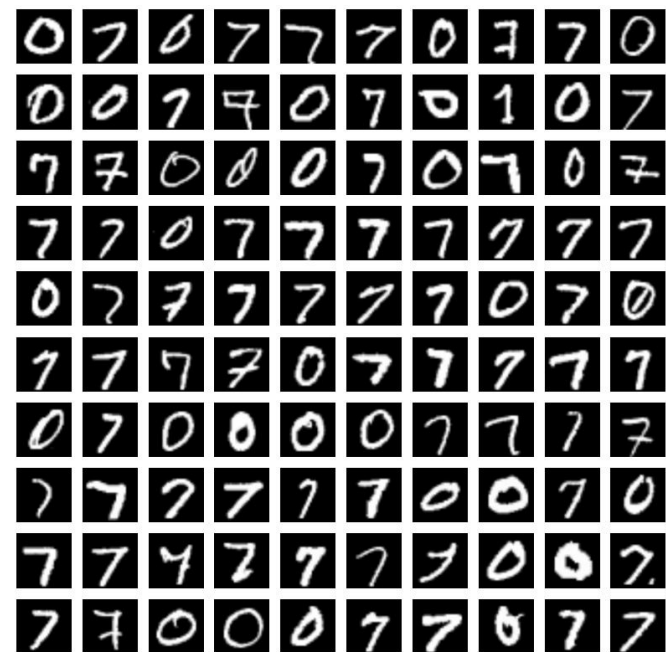
In addition to excluding those inapplicable hyperparameters, based on the features from MNIST images, we modified the input format to adapt the ResNet18 while reducing interpolation noise caused by excessive padding. We used mean pooling as simplified 'distribution' pooling.

Architecture	input - $299 \times 299 \times 3$ ResNet18 (128 nodes in the last fc layer) 'distribution' pooling Dropout(0.5) fc-384 + ReLU Dropout(0.5) fc-192 + ReLU Dropout(0.5) fc-1 (regression)			input - $64 \times 64 \times 3$ ResNet18 (128 nodes in the last fc layer) Mean pooling Dropout (0.5) fc-384 + ReLU Dropout (0.5) fc-192 + ReLU Dropout (0.5) fc-1 (regression)
patch size	512×512			
random crop size	299×299			
# instances per bag (N)	200	# instances per bag		100
# features (J)	128	# features		128
# bins in 'distribution' filters	21	Optimizer		ADAM
σ in Gaussian kernel	0.05	Learning rate		$1e-4$
Optimizer	ADAM	L2 regularization weight decay		0.0005
Learning rate	$1e-4$	batch size		1
L2 regularization weight decay	0.0005			
batch size	1			

Construction of the datasets

We first generated 500 bags containing 100 instances, then generated a uniform distribution of digit 0's fractions between 0 and 1 and disrupted them.

We used `train_test_split` to split the training and test sets, and then randomly picked samples to ensure that the digit 0's fraction of each bag matches the fraction in the distribution and to avoid duplicates.

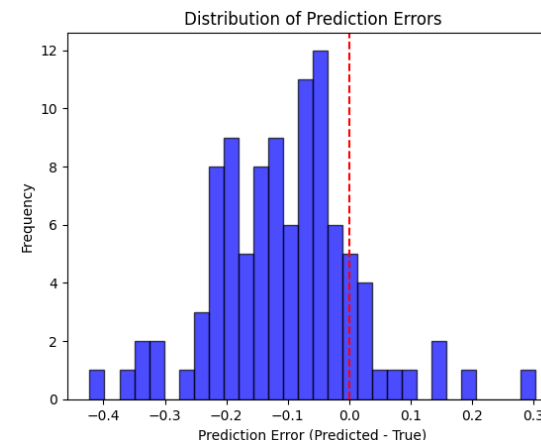
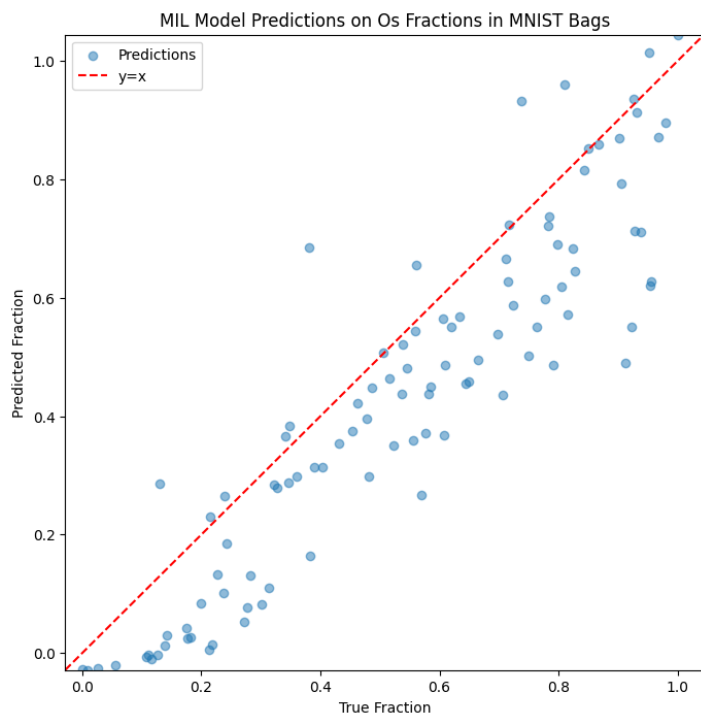


The samples contained within one bag are shown on the top.

The figure on the left shows the distribution of bags with different fractions in the training and test sets. Bags with different fractions are evenly distributed in the training set. The test set has fewer samples but still maintains a similar overall distribution trend.

Model performance and discussion

Below shows a scatter plot of true fraction vs. model-predicted fraction. In general, the model can make prediction roughly close but with systematic undervaluation. The histogram on the right also confirms this and reveals that the error basically obeys the normal distribution. The model gives predictions beyond the $[0,1]$ range in the face of some extreme real fractions (close to 0 or 1).



Since we did not constrain the output of the MLP, the model predicted fractions beyond the range $[0,1]$. The fact that MSE loss does not have penalty for values beyond $[0,1]$ may also be a reason for the distributional shift.

Using Sigmoid to constrain the output of MLP regression, or to normalize the output of the pooling layer, are means to avoid out-of-range prediction when necessary.

As for the systemic undervaluation of the model, it is an option to switch to Smooth L1 or MSLE, which are also suitable for regression tasks. Checking the rationality of mean pooling and trying other pooling methods may also output features for better performance.