Compositional Embeddings for Multi-Label One-Shot Learning: Supplementary Material

A Alternative Training Procedure

We also tried another method of training f and g with the explicit goal of encouraging g to map $e_{\mathcal{T}}$ and $e_{\mathcal{U}}$ to be close to $e_{\mathcal{T}\cup\mathcal{U}}$. This can be done by training f and g alternately, or by training them jointly in the same backpropagation. However, this approach yielded very poor results. A possible explanation is that g could fulfill its goal by mapping all vectors to the same location (e.g., $\mathbf{0}$). Hence, a trade-off arises between g's goal and f's goal (separating examples with distinct label sets).

B Details of Experiment 1: OmniGlot

There are 944 characters in training set, 20 characters in validation set, 659 characters in test set.

To generate a data episode, the rendering function r (1) randomly picks 5 character classes; (2) for each character class randomly selects one image as reference image and one as test image; (3) for each image from previous step applies random affine transformations consisting of shift up to 20%, scaling up to 10%, and rotation up to 10°; (4) generates all possible combinations of 2-sets and 3-sets by taking the minimum value of multiple test images; (5) adds Gaussian noise with mean 0.9 and variance 0.1.

We generate 100,000 episodes for training set, 1,000 episodes for validation set and 10,000 episodes for test set.

Training is performed using Adam (lr = .0003) to maximize the validation accuracy. Every mini-batch contains one data episode. We set the hinge parameter $\epsilon = 0.1$ when computing loss. We do not explore other hyperparameters as our focus is to make comparison of different architectures. The model is trained and evaluated once.

2 students are our university are asked to complete the same task on the first 10 data episodes in the test set. Their results and g_{Lin} are compared in Table 1.

We also conduct another experiment with $|\mathcal{T}| \leq 2$, while other settings are the same. Results can be found in Table 2.

Experiment 1 (OmniGlot) % Correct

	Humans	g_{Lin}
All Exact	71.2	61.2
1-sets Exact	99.0	98.0
2-sets Exact	84.0	58.0
3-sets Exact	44.5	46.0

Table 1: **Experiment 1 (OmniGlot)**: The results on the same test data are compared between 2 students and g_{Lin}.

Experiment 1 (OmniGlot): Train with $|\mathcal{T}| \leq 2$

Label Set Identification							
		$f\&g$ Approaches				Baselines	
		g_{DNN}	$g_{\text{Lin+FC}}$	g_{Lin}	g_{Mean}	Mean	MF
All	Exact	77.8	77.7	81.5	75.3	43.7	6.7
	Top-3	95.4	95.1	96.8	93.8	67.6	20.0
1-sets	Exact	97.4	95.8	96.7	88.0	89.8	6.7
	Top-3	99.5	99.1	99.4	98.3	98.9	20.0
2-sets	Exact	68.0	68.6	73.9	68.9	20.6	6.7
	Top-3	93.3	93.1	95.4	91.5	51.9	20.0
Set Size Determination							
All		96.5	96.6	97.2	90.5	76.7	55.6

Table 2: **Experiment 1 (OmniGlot with $|\mathcal{T}| \leq 2$)**: Mean accuracy (% correct) in inferring the label set of each example exactly (top 1), within the top 3, and the size of each label set. Set Size Determination measures the ability to infer the set size. Models are trained and tested with maximum class set size of 2.

C Details of Experiment 2: OmniGlot

Same as Experiment 1, there are 944 characters in training set, 20 characters in validation set, 659 characters in test set.

To generate a data episode, the rendering function r (1) randomly picks $\mathcal{T}$ character classes ($2 \leq \mathcal{T} \leq 6$); (2) for the $1st$ character class randomly picks one image as positive sample; for each character classes from $1st$ to $(\mathcal{T}-1)th$ randomly picks one image as singleton candidate; for the $\mathcal{T}th$ character class randomly picks one image as negative sample; (3) for each image from previous step applies random affine transformations consisting of shift up to 20%, scaling up to 10%, and rotation up to 10°; (4) generates the compositional image by taking the minimum value of the singleton candidates; (5) adds Gaussian noise with mean 0.9 and variance 0.1.

We generate 100,000 episodes for training set, 1,000 episodes for validation

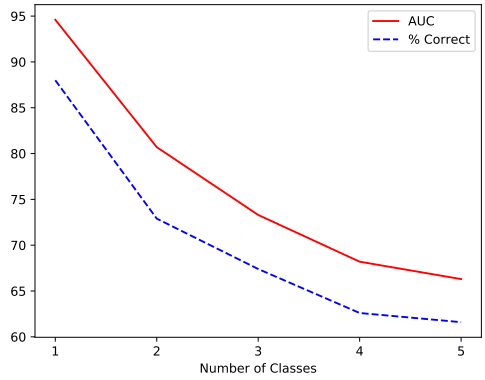

Figure 1: In OmniGlot, h_{DNN}'s results according to the number of subclasses contained in images

set and 10,000 episodes for test set.

Unlike Experiment 1, the symmetric function of h's first layer is replaced by (1) $W_1 a + W_2 b$ in g_{Lin}, $g_{\mathrm{Lin+FC}}$ and g_{DNN}; (2) $W\mathrm{Cat}(a, b)$ in g_{Mean}, where $\mathrm{Cat}(a, b)$ is concatenation of a and b. The output dimension of each h's last layer is modified to 1.

Training is performed using Adam (lr = .0003) to maximize the validation accuracy. Every mini-batch contains 128 data episode. Binary cross entropy is used as loss function. The model is trained and evaluated once.

Additionally, we also plot the relationship between accuracy/AUC and number of singletons in compositional sample ($\mathcal{T}$). See in Figure 1.

D Details of Experiment 3: Open Images

In Open Images, there are 1,743,042 training images, 41,620 validation images and 125,436 test images. There are 600 classes of objects contained in these images in total. In order to make sure that object classes in evaluation are not seen during training, 500 classes are used for training (validation set also uses the same 500 classes) and 73 classes are used for testing (Not all 600 classes are included in test set).

In proposed method and TradEmb baseline, all objects are cropped according to their bounding boxes, and then resized and padded to 256×256. All orignial images are also resized and padded to the same size.

In SlideWin baseline, all test images are orginial images instead of resizing to 256×256. Because down sampling would make the image quality of sliding windows to be too bad for recognition. The architecture used in this baseline is ResNet18 with output dimension modified to 2.

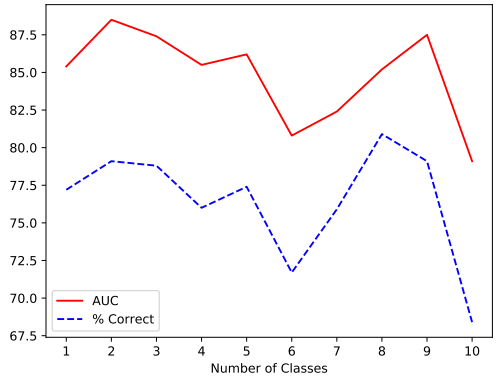

Figure 2: In Open Images, h_{DNN}'s results according to the number of subclasses contained in images: Results of images contain more than 10 labeled objects are not show because they are too few in test set.

We also generate two mapping dictionaries: (1) images and the classes of the objects they contained; (2) object classes and all images that contain them.

To generate a data episode, we (1) randomly pick one class as positive class; (2) randomly pick one test image that contains the positive class; (3) randomly pick one negative class that is not contained in the test image; (4) randomly pick one object image from positive class and one from negative class.

We generate 100,000 episodes for training set, 1,000 episodes for validation set and 10,000 episodes for test set.

In this experiment, function h is the same as Experiment 2. f is ResNet pretrained on ImageNet.

Training is performed using Adam ($\text{lr} = 3 \times 10^{-8}$ for f, $\text{lr} = 3 \times 10^{-8}$ for h) to maximize the validation accuracy. Every mini-batch contains 32 data episode. Binary cross entropy is used as loss function. The model is trained and evaluated once.

Additionally, we also plot the relationship between accuracy/AUC and number of singletons in compositional sample ($\mathcal{T}$). See in Figure 2

E Additional Results

The computational complexity and number of parameters are shown in Table 3.

Figure 3 shows the all exact % Correct and standard deviation of different f&g models in Experiment 1 and Experiment 2.

All experiments are conducted on one NVIDIA TITAN RTX and one NVIDIA GEFORCE GTX 1080 Ti.

Model Complexity Comparison

Model I				
	g_{DNN}	$g_{\text{Lin+FC}}$	g_{Lin}	TradEmb
MACs	12,480	8,256	4,096	0
Params	5,472	3,232	2,080	0
Model II				
	h_{DNN}	$h_{\text{Lin+FC}}$	h_{Lin}	TradEmb
MACs	8,448	4,224	128	0
Params	4,449	2,209	65	0

Table 3: **Model Complexity**: Top table shows the number of multiply–accumulate operations and parameters of g functions when the embedding dimension is 32. Bottom table shows the number of multiply–accumulate operations and parameters of h functions when the embedding dimension is 32. TradEmb has none of both because it only uses f function.

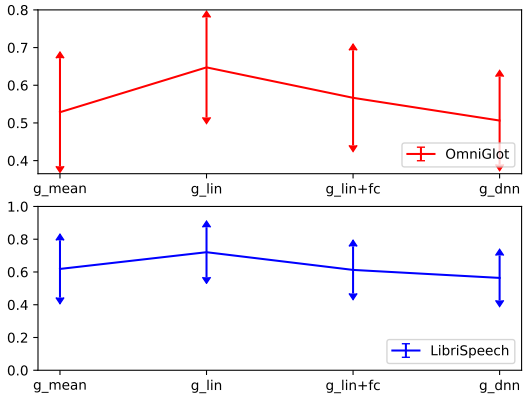

Figure 3: **% Correct and standard deviation:** Shows the all exact % Correct of different f&g models. Error bar shows the standard deviation of accuracy in all test data episodes.