

APPENDIX

Anonymous authors

Paper under double-blind review

ABSTRACT

In our supplementary materials we provide implementation details, additional experimental results and further exploration of our latent embeddings. We also present our results for the lesser addressed Lake split (Lake et al., 2015) of the Omniglot few-shot classification benchmark, detail on our differentiable rasterization function f_{raster} and our data processing procedures.

A RASTERIZATION

The key enabler of our novel pixel loss for sketch drawings is our differentiable rasterization function f_{raster} . Sequence based loss functions such as $\mathcal{L}_{\text{stroke}}$ are sensitive to the order of points while in reality, drawings are sequence invariant. Visually, a square is a square whether it is drawn clockwise or counterclockwise.

The purpose of a sketch representation is to lower the complexity of the data space and decode in a more visually intuitive manner. While it is a necessary departure point, the sequential generation of drawings is not key to our visual representation and we would like SketchEmbedNet to be agnostic to any specific sequence needed to draw the sketch that is representative of the image input.

To facilitate this, we develop our rasterization function f_{raster} which renders an input sequence of strokes as a pixel image. However, during training, the RNN outputs a mixture of Gaussians at each timestep. To convert this to a stroke sequence, we sample from these Gaussians; this can be repeated to reduce the variance of the pixel loss. We then scale our predicted and ground truth sequences by the properties of the latter before rasterization.

Stroke sampling At the end of sequence generation we have $N_s \times (6M + 3)$ parameters, 6 Gaussian mixture parameters, 3 pen states, N_s times, one for each stroke. To obtain the actual drawing we sample from the mixture of Gaussians:

$$\begin{bmatrix} \Delta x_t \\ \Delta y_t \end{bmatrix} = \begin{bmatrix} \mu_{x,t} \\ \mu_{y,t} \end{bmatrix} + \begin{bmatrix} \sigma_{x,t} & 0 \\ \rho_{xy,t}\sigma_{y,t} & \sigma_{y,t}\sqrt{1-\rho_{xy,t}^2} \end{bmatrix} \epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{1}_2). \quad (1)$$

After sampling we compute the cumulative sum of every stroke over the timestep so that we obtain the absolute displacement from the initial position:

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \sum_{\tau=0}^T \begin{bmatrix} \Delta x_\tau \\ \Delta y_\tau \end{bmatrix}. \quad (2)$$

$$\mathbf{y}_{t,abs} = (x_t, y_t, s_1, s_2, s_3). \quad (3)$$

Scaling Each sketch generated by our model begins at (0,0) and the variance of all strokes in the training set is normalized to 1. On a fixed canvas the image is both very small and localized to the top left corner. We remedy this by computing a scale λ and shift $x_{\text{shift}}, y_{\text{shift}}$ using labels \mathbf{y} and apply them to both the prediction \mathbf{y}' as well as the ground truth \mathbf{y} . These parameters are computed as:

$$\lambda = \min \left\{ \frac{W}{x_{\max} - x_{\min}}, \frac{H}{y_{\max} - y_{\min}} \right\}, \quad (4)$$

$$x_{\text{shift}} = \frac{x_{\max} + x_{\min}}{2} \lambda, \quad y_{\text{shift}} = \frac{y_{\max} + y_{\min}}{2} \lambda. \quad (5)$$

$x_{\max}, x_{\min}, y_{\max}, y_{\min}$ are the minimum and maximum values of x_t, y_t from the supervised stroke labels and not the generated strokes. W and H are the width and height in pixels of our output canvas.

Calculate pixel intensity Finally we are able to calculate the pixel p_{ij} intensity of every pixel in our $H \times W$ canvas.

$$p_{ij} = \sigma \left[2 - 5 \times \min_{t=1 \dots N_s} \left(\text{dist}((i, j), (x_{t-1}, y_{t-1}), (x_t, y_t)) + (1 - \lfloor s_{1,t-1} \rfloor) 10^6 \right) \right], \quad (6)$$

where the distance function is the distance between point (i, j) from the line segment defined by the absolute points (x_{t-1}, y_{t-1}) and (x_t, y_t) . We also blow up any distances where $s_{1,t-1} < 0.5$ so as to not render any strokes where the pen is not touching the paper.

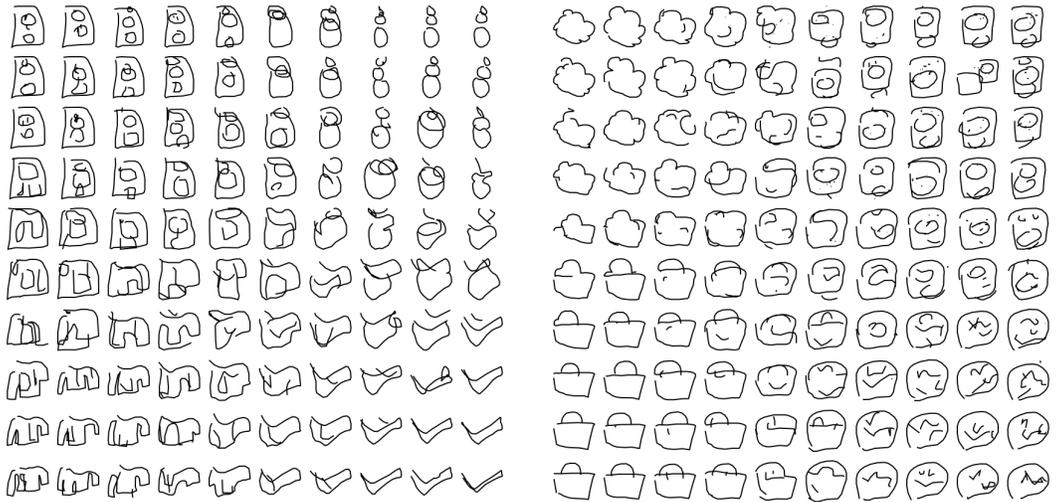
B IMPLEMENTATION DETAILS

We train our model for 300k iterations with a batch size of 256 for the Quickdraw dataset and 64 for Sketchy due to memory constraints. The initial learning rate is 1e-3 which decays by 0.85 every 15k steps. We use the Adam (Kingma & Ba, 2015) optimizer and clip gradient values at 1.0. $\sigma = 2.0$ is used for the Gaussian blur in $\mathcal{L}_{\text{pixel}}$. For the curriculum learning schedule, the value of α is set to 0 initially and increases by 0.05 every 10k training steps with an empirically obtained cap at $\alpha_{\text{max}} = 0.50$ for Quickdraw and $\alpha_{\text{max}} = 0.75$ for Sketchy.

The ResNet12 (Oreshkin et al., 2018) encoder uses 4 ResNet blocks with 64, 128, 256, 512 filters respectively and ReLU activations. The Conv4 backbone has 4 blocks of convolution, batch norm (Ioffe & Szegedy, 2015), ReLU and max pool, identical to Vinyals et al. (2016). We select the latent space to be 256 dimensions, RNN output size to be 1024, and the hypernetwork embedding size to be 64. We use a mixture of $M = 30$ bivariate Gaussians for the mixture density output of the stroke offset distribution.

C LATENT SPACE INTERPOLATION

Like in many encoding-decoding models we evaluate the interpolation of our latent space. We select 4 embeddings at random and use bi-linear interpolation to produce new embeddings. Results are in Figures 1a and 1b.



(a) Interpolation of classes: power outlet, snowman, jacket, elbow

(b) Interpolation of classes: cloud, power outlet, basket, compass

Figure 1: Latent space interpolations of randomly selected examples

We observe that compositionality is also present in these interpolations. In the top row of Figure 1a, the model first plots a third small circle when interpolating from the 2-circle power outlet and the 3-circle snowman. This small circle is treated as single component that grows as it transitions between classes until it's final size in the far right snowman drawing.

Some other RNN-based sketching models (Ha & Eck, 2018; Chen et al., 2017) experience other classes materializing in interpolations between two unrelated classes. Our model does not exhibit this same behaviour as our embedding space is learned from more classes and thus does not contain local groupings of classes.

D EFFECT OF α ON FEW-SHOT CLASSIFICATION

We performed additional experiments exploring the impact of our curriculum training schedule for α . The encoding component of our drawing model was evaluated on the few-shot classification task for different values of α_{\max} every 25k iterations during training. A graph is shown in Figure 2 and the full table of all values of α_{\max} is in Table 1.

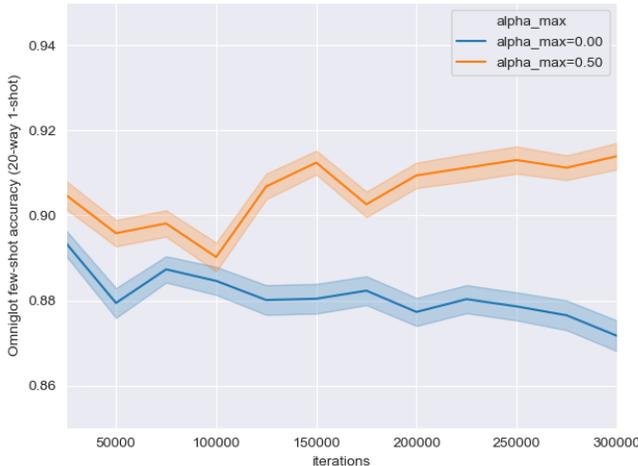


Figure 2: Few-shot classification accuracy of α_{\max} values 0.0 and 0.5 over training

Table 1: Few-shot classification accuracy of all α_{\max} values

α_{\max}	25k	50k	75k	100k	125k	150k	175k	200k	225k	250k	275k	300k
0.00	89.35	87.94	88.73	88.46	88.01	88.04	88.23	87.73	88.03	87.86	87.65	87.17
0.25	89.21	90.39	90.20	89.75	87.78	88.37	88.64	88.05	87.98	88.41	88.15	87.82
0.50	90.48	89.58	89.81	89.02	90.68	91.24	90.26	90.94	91.12	91.30	91.12	91.39
0.75	91.39	89.95	89.56	89.81	89.95	90.79	91.02	91.09	91.82	90.76	91.42	90.59
0.95	90.23	90.15	90.10	89.55	90.27	92.37	92.27	90.29	91.58	91.02	89.73	89.77

E INTRA-ALPHABET LAKE SPLIT

The creators of the Omniglot dataset and one-shot classification benchmark originally proposed an intra-alphabet classification task. This task is more challenging than the common Vinyals split as characters from the same alphabet may exhibit similar stylistics of sub-components that makes visual differentiation more difficult. This benchmark has been less explored by researchers; however, we still present the performance of our SketchEmbedding model against evaluations of other few-shot classification models on the benchmark. Results are shown in Table 2.

Table 2: Few-shot classification results on Omniglot (Lake split)

Algorithm	Omniglot (Lake split)		(way, shot)			
	Backbone	Train Data	(5,1)	(5,5)	(20,1)	(20,5)
Conv-VAE	Conv4	Quickdraw	73.12 \pm 0.58	88.50 \pm 0.39	53.45 \pm 0.51	73.62 \pm 0.48
SketchEmbedding (<i>Ours</i>)	Conv4	Quickdraw	89.16 \pm 0.41	97.12 \pm 0.18	74.24 \pm 0.48	89.87 \pm 0.25
SketchEmbedding (<i>Ours</i>)	ResNet12	Quickdraw	91.03 \pm 0.37	97.91 \pm 0.15	77.94 \pm 0.44	92.49 \pm 0.21
BPL (<i>Supervised</i>) (Lake et al., 2015; 2019)	N/A	Omniglot	-	-	96.70	-
ProtoNet (<i>Supervised</i>) (Snell et al., 2017; Lake et al., 2019)	Conv4	Omniglot	-	-	86.30	-
RCN (<i>Supervised</i>) (George et al., 2017; Lake et al., 2019)	N/A	Omniglot	-	-	92.70	-
VHE (<i>Supervised</i>) (Hewitt et al., 2018; Lake et al., 2019)	N/A	Omniglot	-	-	81.30	-

Unsurprisingly, our model is outperformed by supervised models and does fall behind by a more substantial margin than in the Vinyals split. However, our SketchEmbedding approach still achieves respectable classification accuracy overall and greatly outperforms a Conv-VAE baseline.

F EFFECT OF RANDOM SEEDING ON FEW-SHOT CLASSIFICATION

The training objective for SketchEmbedNet is to reproduce sketch drawings of the input. This task is unrelated to few-shot classification may perform variably given different initialization. We quantify this variance by training our model with 15 unique random seeds and evaluating the performance of the latent space on the few-shot classification tasks.

We disregard the per (evaluation) episode variance of our model in each test stage and only present the mean accuracy. We then compute a new confidence interval over random seeds. Results are presented in Tables 3, 4, 5.

Table 3: Random Seeding on Few-Shot Classification results on Omniglot (Conv4)

Seed	(way, shot)			
	(5,1)	(5,5)	(20,1)	(20,5)
1	96.45	99.41	90.84	98.08
2	96.54	99.48	90.82	98.10
3	96.23	99.40	90.05	97.94
4	96.15	99.46	90.50	97.99
5	96.21	99.40	90.54	98.10
6	96.08	99.43	90.20	97.93
7	96.19	99.39	90.70	98.05
8	96.68	99.44	91.11	98.18
9	96.49	99.42	90.64	98.06
10	96.37	99.47	90.50	97.99
11	96.52	99.40	91.13	98.18
12	96.96	99.50	91.67	98.30
13	96.31	99.38	90.57	98.04
14	96.12	99.45	90.54	98.03
15	96.30	99.48	90.62	98.05
Average	96.37 \pm 0.12	99.43 \pm 0.02	90.69 \pm 0.20	98.07 \pm 0.05

Table 4: Random Seeding on Few-Shot Classification results on Omniglot (ResNet12)

(way, shot)				
Seed	(5,1)	(5,5)	(20,1)	(20,5)
1	96.61	99.58	91.25	98.58
2	96.37	99.52	90.44	98.40
3	96.04	99.58	89.86	98.27
4	96.44	99.50	90.76	98.40
5	95.95	99.52	89.88	98.29
6	95.63	99.45	89.28	98.17
7	96.24	99.52	89.90	98.34
8	95.41	99.45	88.75	98.05
9	96.04	99.49	89.70	98.24
10	95.40	99.41	88.91	98.05
11	95.82	99.51	89.67	98.24
12	96.25	99.51	90.21	98.28
13	95.84	99.53	89.71	98.18
14	96.04	99.56	89.89	98.31
15	96.04	99.57	89.97	98.32
Average	96.00 ± 0.31	99.51 ± 0.04	89.89 ± 0.56	98.27 ± 0.12

Table 5: Random Seeding on Few-Shot Classification results on mini-ImageNet

(way, shot)				
Seed	(5,1)	(5,5)	(5,20)	(5,50)
1	37.15	52.99	63.92	68.72
2	39.38	55.20	65.60	69.79
3	39.40	55.47	65.94	70.41
4	40.39	57.15	67.60	71.99
5	38.40	54.08	65.36	70.08
6	37.94	53.98	65.24	69.65
7	38.88	55.71	66.59	71.35
8	37.89	52.65	63.42	68.14
9	38.25	53.86	65.02	69.82
10	39.11	55.29	65.99	69.98
11	37.39	52.88	63.66	68.33
12	38.24	53.91	65.19	69.82
13	38.62	53.84	63.83	68.69
14	37.73	53.61	64.22	68.41
15	39.50	55.23	65.51	70.25
Average	38.55 ± 0.45	54.39 ± 0.63	65.14 ± 0.59	69.69 ± 0.56

G DATA PROCESSING

G.1 QUICKDRAW

We apply the same data processing methods as in Ha & Eck (2018) with no additional changes to produce our stroke labels \mathbf{y} . When rasterizing for our input \mathbf{x} , we scale, center the strokes then pad the image with 10% of the resolution in that dimension rounded to the nearest integer.

The following list of classes were used for training: The Eiffel Tower, The Mona Lisa, aircraft carrier, alarm clock, ambulance, angel, animal migration, ant, apple, arm, asparagus, banana, barn, baseball, baseball bat, bathtub, beach, bear, bed, bee, belt, bench, bicycle, binoculars, bird, blueberry, book, boomerang, bottlecap, bread, bridge, broccoli, broom, bucket, bulldozer, bus, bush, butterfly, cactus, cake, calculator, calendar, camel, camera, camouflage, campfire, candle, cannon, car, carrot, castle, cat, ceiling fan, cell phone, cello, chair, chandelier, church, circle, clarinet, clock, coffee cup, computer, cookie, couch, cow, crayon, crocodile, crown, cruise ship, diamond, dishwasher, diving board, dog, dolphin, donut, door, dragon, dresser, drill, drums, duck, dumbbell, ear, eye, eyeglasses, face, fan, feather, fence, finger, fire hydrant, fireplace, firetruck, fish, flamingo, flashlight, flip flops, flower, foot, fork, frog, frying pan, garden, garden hose, giraffe, goatee, grapes, grass, guitar, hamburger, hand, harp, hat, headphones, hedgehog, helicopter, helmet, hockey puck, hockey stick, horse, hospital, hot air balloon, hot dog, hourglass, house, house plant, ice cream, key, keyboard, knee, knife, ladder, lantern, leaf, leg, light bulb, lighter, lighthouse, lightning, line, lipstick, lobster, mailbox, map, marker, matches, megaphone, mermaid, microphone, microwave, monkey, mosquito, motorbike, mountain, mouse, moustache, mouth, mushroom, nail, necklace, nose, octopus, onion, oven, owl, paint can, paintbrush, palm tree, parachute, passport, peanut, pear, pencil, penguin, piano, pickup truck, pig, pineapple, pliers, police car, pool, popsicle, postcard, purse, rabbit, raccoon, radio, rain, rainbow, rake, remote control, rhinoceros, river, rollerskates, sailboat, sandwich, saxophone, scissors, see saw, shark, sheep, shoe, shorts, shovel, sink, skull, sleeping bag, smiley face, snail, snake, snowflake, soccer ball, speedboat, square, star, steak, stereo, stitches,

stop sign, strawberry, streetlight, string bean, submarine, sun, swing set, syringe, t-shirt, table, teapot, teddy-bear, tennis racquet, tent, tiger, toe, tooth, toothpaste, tractor, traffic light, train, triangle, trombone, truck, trumpet, umbrella, underwear, van, vase, watermelon, wheel, windmill, wine bottle, wine glass, wristwatch, zigzag, blackberry, power outlet, peas, hot tub, toothbrush, skateboard, cloud, elbow, bat, pond, compass, elephant, hurricane, jail, school bus, skyscraper, tornado, picture frame, lollipop, spoon, saw, cup, roller coaster, pants, jacket, rifle, yoga, toilet, waterslide, axe, snowman, bracelet, basket, anvil, octagon, washing machine, tree, television, bowtie, sweater, backpack, zebra, suitcase, stairs, The Great Wall of China

G.2 OMNIGLOT

We derive our Omniglot tasks from the stroke dataset originally provided by Lake et al. (2015) rather than the image analogues. We translate the Omniglot stroke-by-stroke format to the same one used in Quickdraw. Then we apply the Ramer-Douglas-Peucker (Douglas & Peucker, 1973) algorithm with an epsilon value of 2 and normalize variance to 1 to produce y . We also rasterize our images in the same manner as above for our input x .

G.3 SKETCHY

Sketchy data is provided as an SVG image composed of line paths that are either straight lines or Bezier curves. To generate stroke data we sample sequences of points from Bezier curves at a high resolution that we then simplify with RDP, $\epsilon = 5$. We also eliminate continuous strokes with a short path length or small displacement to reduce our stroke length and remove small and noisy strokes. Path length and displacement are considered with respect to the scale of the entire sketch.

Once again we normalize stroke variance and rasterize for our input image in the same manners as above.

The following classes were use for training after removing overlapping classes with mini-ImageNet: hot-air_balloon, violin, tiger, eyeglasses, mouse, jack-o-lantern, lobster, teddy_bear, teapot, helicopter, duck, wading_bird, rabbit, penguin, sheep, windmill, piano, jellyfish, table, fan, beetle, cabin, scorpion, scissors, banana, tank, umbrella, crocodilian, volcano, knife, cup, saxophone, pistol, swan, chicken, sword, seal, alarm_clock, rocket, bicycle, owl, squirrel, hermit_crab, horse, spoon, cow, hotdog, camel, turtle, pizza, spider, songbird, rifle, chair, starfish, tree, airplane, bread, bench, harp, seagull, blimp, apple, geyser, trumpet, frog, lizard, axe, sea_turtle, pretzel, snail, butterfly, bear, ray, wine_bottle, , elephant, raccoon, rhinoceros, door, hat, deer, snake, ape, flower, car_(sedan), kangaroo, dolphin, hamburger, castle, pineapple, saw, zebra, candle, cannon, racket, church, fish, mushroom, strawberry, window, sailboat, hourglass, cat, shoe, hedgehog, couch, giraffe, hammer, motorcycle, shark

H AUTOREGRESSIVE DRAWING MODEL COMPARISONS

We summarize the key components of SketchEmbedNet in comparison to other autoregressive drawing models in Table 6.

Table 6: Model comparisons between generative autoregressive models that produce pixel or vector sketch drawings.

Autoregressive sketching models					
Model	Dataset	# classes	Encoder	Decoder	Loss function
Handwriting Sequence Graves (2013)	IAM-OnDB Liwicki & Bunke (2005)	1	RNN	Mixture Density RNN	\mathcal{L}_{stroke}
DRAW Gregor et al. (2015)	SVHNNetzer et al. (2011), MNIST LeCun et al. (1998)	10	RNN	RNN	$\mathcal{L}_{pixel} + \mathcal{L}_{KL}$
Sketch-RNN Ha & Eck (2018)	Quickdraw Jongejan et al. (2016)	1	Bi-directional RNN	Mixture Density RNN	$\mathcal{L}_{pen} + \mathcal{L}_{stroke} + \mathcal{L}_{KL}$
Sketch-pix2seq Chen et al. (2017)	Quickdraw Jongejan et al. (2016)	3, 6	simple CNN	Mixture Density RNN	$\mathcal{L}_{pen} + \mathcal{L}_{stroke}$
AI-Sketcher Cao et al. (2019)	Quickdraw Jongejan et al. (2016), FaceX Cao et al. (2019)	5, 10, 15, 20	Bi-directional RNN + CNN Autoencoder	Mixture Density RNN	$\mathcal{L}_{pen} + \mathcal{L}_{stroke} + \mathcal{L}_{KL}$
deep_p2s Song et al. (2018)	Quickdraw Jongejan et al. (2016), ShoesV2 Yu et al. (2016), ChairV2	1	Bi-directional RNN, CNN	CNN, Mixture Density RNN	$\mathcal{L}_{pen} + \mathcal{L}_{stroke} + \mathcal{L}_{l2} + \mathcal{L}_{KL} + \mathcal{L}_{shortcut}$
SketchEmbedding (ours)	Quickdraw Jongejan et al. (2016)	300	ResNet12 Oreshkin et al. (2018)	Mixture Density RNN	$\mathcal{L}_{pen} + \mathcal{L}_{stroke} + \mathcal{L}_{pixel}$

I FEW-SHOT CLASSIFICATION ON OMNIGLOT – FULL RESULTS

The full results table for few-shot classification on the Omniglot (Lake et al., 2015) dataset, including the ResNet12 (Oreshkin et al., 2018) model.

Table 7: Few-shot classification results on Omniglot

Omniglot			(way, shot)			
Algorithm	Backbone	Train Data	(5,1)	(5,5)	(20,1)	(20,5)
Training from Scratch (Hsu et al., 2019)	N/A	Omniglot	52.50 ± 0.84	74.78 ± 0.69	24.91 ± 0.33	47.62 ± 0.44
Random CNN	Conv4	N/A	67.96 ± 0.44	83.85 ± 0.31	44.39 ± 0.23	60.87 ± 0.22
Conv-VAE	Conv4	Omniglot	77.83 ± 0.41	92.91 ± 0.19	62.59 ± 0.24	84.01 ± 0.15
Conv-VAE	Conv4	Quickdraw	81.49 ± 0.39	94.09 ± 0.17	66.24 ± 0.23	86.02 ± 0.14
Conv-AE	Conv4	Quickdraw	81.54 ± 0.40	93.57 ± 0.19	67.24 ± 0.24	84.15 ± 0.16
β -VAE ($\beta = 250$) (Higgins et al., 2017)	Conv4	Quickdraw	79.11 ± 0.40	93.23 ± 0.19	63.67 ± 0.24	84.92 ± 0.15
k-NN (Hsu et al., 2019)	N/A	Omniglot	57.46 ± 1.35	81.16 ± 0.57	39.73 ± 0.38	66.38 ± 0.36
Linear Classifier (Hsu et al., 2019)	N/A	Omniglot	61.08 ± 1.32	81.82 ± 0.58	43.20 ± 0.69	66.33 ± 0.36
MLP + Dropout (Hsu et al., 2019)	N/A	Omniglot	51.95 ± 0.82	77.20 ± 0.65	30.65 ± 0.39	58.62 ± 0.41
Cluster Matching (Hsu et al., 2019)	N/A	Omniglot	54.94 ± 0.85	71.09 ± 0.77	32.19 ± 0.40	45.93 ± 0.40
CACTUs-MAML (Hsu et al., 2019)	Conv4	Omniglot	68.84 ± 0.80	87.78 ± 0.50	48.09 ± 0.41	73.36 ± 0.34
CACTUs-ProtoNet (Hsu et al., 2019)	Conv4	Omniglot	68.12 ± 0.84	83.58 ± 0.61	47.75 ± 0.43	66.27 ± 0.37
AAL-ProtoNet (Antoniou & Storkey, 2019)	Conv4	Omniglot	84.66 ± 0.70	88.41 ± 0.27	68.79 ± 1.03	74.05 ± 0.46
AAL-MAML (Antoniou & Storkey, 2019)	Conv4	Omniglot	88.40 ± 0.75	98.00 ± 0.32	70.20 ± 0.86	88.30 ± 1.22
UMTRA (Khodadadeh et al., 2019)	Conv4	Omniglot	83.80	95.43	74.25	92.12
SketchEmbedding (<i>Ours</i>)	Conv4	Omniglot	94.88 ± 0.22	99.01 ± 0.08	86.18 ± 0.18	96.69 ± 0.07
SketchEmbedding-avg (<i>Ours</i>)	Conv4	Quickdraw	96.37	99.43	90.69	98.07
SketchEmbedding-best (<i>Ours</i>)	Conv4	Quickdraw	96.96 ± 0.17	99.50 ± 0.06	91.67 ± 0.14	98.30 ± 0.05
SketchEmbedding-avg (<i>Ours</i>)	ResNet12	Quickdraw	96.00	99.51	89.88	98.27
SketchEmbedding-best (<i>Ours</i>)	ResNet12	Quickdraw	96.61 ± 0.19	99.58 ± 0.06	91.25 ± 0.15	98.58 ± 0.05
SketchEmbedding(KL)-avg (<i>Ours</i>)	Conv4	Quickdraw	96.06	99.40	89.83	97.92
SketchEmbedding(KL)-best (<i>Ours</i>)	Conv4	Quickdraw	96.60 ± 0.18	99.46 ± 0.06	90.84 ± 0.15	98.09 ± 0.06
SketchEmbedding (<i>w/ Labels</i>) (<i>Ours</i>)	Conv4	Quickdraw	88.52 ± 0.34	96.73 ± 0.13	71.35 ± 0.24	88.16 ± 0.14
MAML (<i>Supervised</i>) (Finn et al., 2017)	Conv4	Omniglot	94.46 ± 0.35	98.83 ± 0.12	84.60 ± 0.32	96.29 ± 0.13
ProtoNet (<i>Supervised</i>) (Snell et al., 2017)	Conv4	Omniglot	98.35 ± 0.22	99.58 ± 0.09	95.31 ± 0.18	98.81 ± 0.07

* Stroke data used for training

J FEW-SHOT CLASSIFICATION ON MINI-IMAGENET – FULL RESULTS

The full results table for few-shot classification on the mini-ImageNet dataset, including the ResNet12 (Oreshkin et al., 2018) model and Conv4 models.

Table 8: Few-shot classification results on mini-ImageNet

mini-ImageNet			(way, shot)			
Algorithm	Backbone	Train Data	(5,1)	(5,5)	(5,20)	(5,50)
Training from Scratch (Hsu et al., 2019)	N/A	mini-ImageNet	27.59 ± 0.59	38.48 ± 0.66	51.53 ± 0.72	59.63 ± 0.74
UMTRA (Khodadadeh et al., 2019)	Conv4	mini-ImageNet	39.93	50.73	61.11	67.15
CACTUs-MAML (Hsu et al., 2019)	Conv4	mini-ImageNet	39.90 ± 0.74	53.97 ± 0.70	63.84 ± 0.70	69.64 ± 0.63
CACTUs-ProtoNet (Hsu et al., 2019)	Conv4	mini-ImageNet	39.18 ± 0.71	53.36 ± 0.70	61.54 ± 0.68	63.55 ± 0.64
AAL-ProtoNet (Antoniou & Storkey, 2019)	Conv4	mini-ImageNet	37.67 ± 0.39	40.29 ± 0.68	-	-
AAL-MAML (Antoniou & Storkey, 2019)	Conv4	mini-ImageNet	34.57 ± 0.74	49.18 ± 0.47	-	-
Random CNN	Conv4	N/A	26.85 ± 0.31	33.37 ± 0.32	38.51 ± 0.28	41.41 ± 0.28
Conv-VAE	Conv4	mini-ImageNet	23.30 ± 0.21	26.22 ± 0.20	29.93 ± 0.21	32.57 ± 0.20
Conv-VAE	Conv4	Sketchy	23.27 ± 0.18	26.28 ± 0.19	30.41 ± 0.19	33.97 ± 0.19
Random CNN	ResNet12	N/A	28.59 ± 0.34	35.91 ± 0.34	41.31 ± 0.33	44.07 ± 0.31
Conv-VAE	ResNet12	mini-ImageNet	23.82 ± 0.23	28.16 ± 0.25	33.64 ± 0.27	37.81 ± 0.27
Conv-VAE	ResNet12	Sketchy	24.61 ± 0.23	28.85 ± 0.23	35.72 ± 0.27	40.44 ± 0.28
SketchEmbedding-avg (<i>ours</i>)	Conv4	Sketchy*	37.01	51.49	61.41	65.75
SketchEmbedding-best (<i>ours</i>)	Conv4	Sketchy*	38.61 ± 0.42	53.82 ± 0.41	63.34 ± 0.35	67.22 ± 0.32
SketchEmbedding-avg (<i>ours</i>)	ResNet12	Sketchy*	38.55	54.39	65.14	69.70
SketchEmbedding-best (<i>ours</i>)	ResNet12	Sketchy*	40.39 ± 0.44	57.15 ± 0.38	67.60 ± 0.33	71.99 ± 0.3
MAML (<i>supervised</i>) (Finn et al., 2017)	Conv4	mini-ImageNet	46.81 ± 0.77	62.13 ± 0.72	71.03 ± 0.69	75.54 ± 0.62
ProtoNet (<i>supervised</i>) (Snell et al., 2017)	Conv4	mini-ImageNet	46.56 ± 0.76	62.29 ± 0.71	70.05 ± 0.65	72.04 ± 0.60

* Stroke data used for training

K ADDITIONAL CONCEPTUAL COMPOSITIONALITY

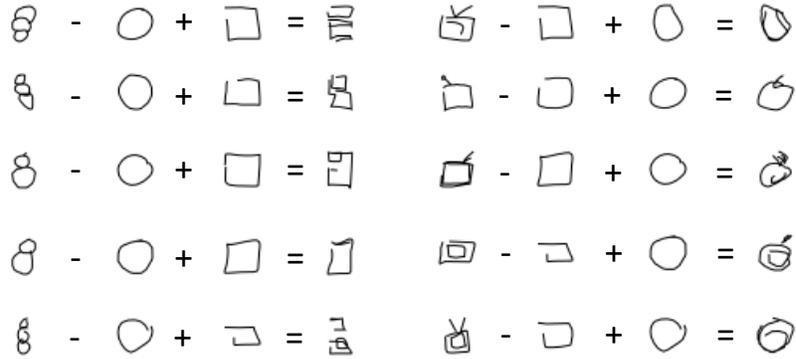
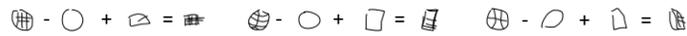


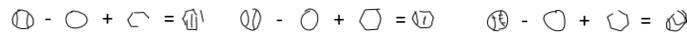
Figure 3: Uncherrypicked conceptual compositionality examples

Conceptual Compositionality Examples

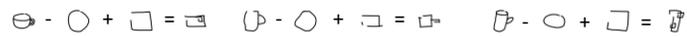
Basketball - Circle + Square



Baseball - Circle + Hexagon



Cup - Circle + Square



Envelope - Square + Hexagon

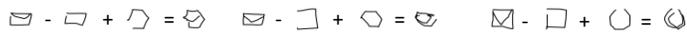


Figure 4: Additional conceptual compositionality examples

L EMBEDDING PROPERTIES OF OTHER BASELINE MODELS

Here we substantiate the uniqueness of the properties observed in SketchEmbeddings by applying the same experiments to a β -VAE (Higgins et al., 2017) as well a vanilla autoencoder trained on the same dataset. We also include results of a SketchEmbedNet trained with a KL objective.

L.1 β -VAE

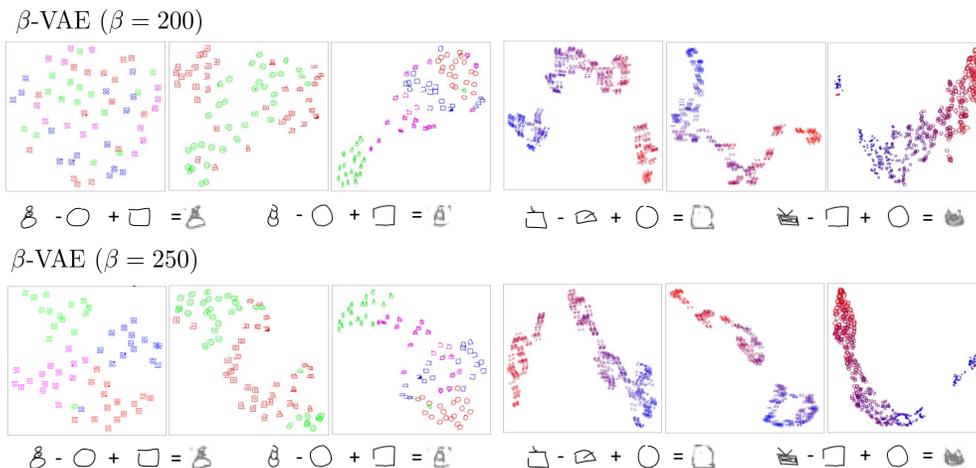


Figure 5: Section ?? results for β -VAE

The β -VAE (Higgins et al., 2017) exhibits similar unsupervised clustering in comparison to the Conv-VAE and is generally incapable of distinguishing input images that have different shape compositions but the same overall silhouette (first two examples from the left). Differently it is better at distinguishing non-synthetic examples that contain multiple squares or circles (3rd figure). However, it utterly fails the latent variable regression task and does not exhibit any significant conceptual composition in latent space.

L.2 AUTOENCODER AND SKETCHEMBEDNET-KL

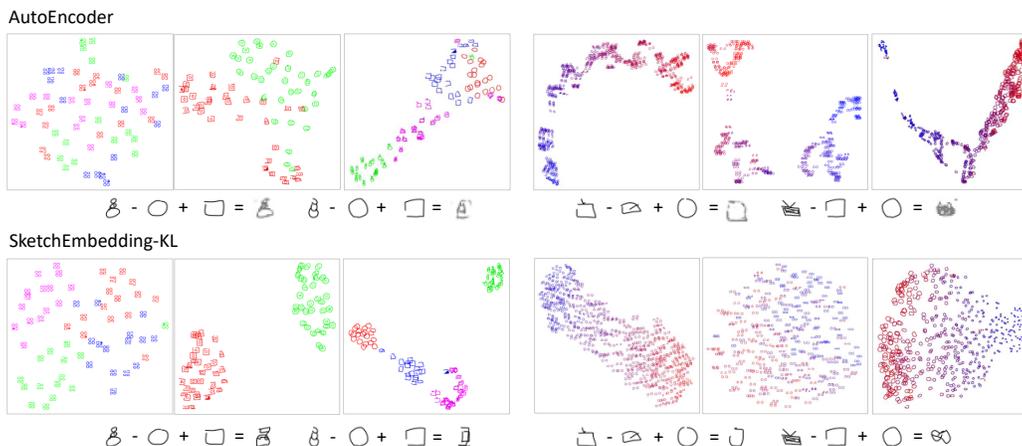


Figure 6: Section ?? results for AutoEncoder and SketchEmbedding-KL

We show that the performance of SketchEmbedding embeddings in our experiments in Section ?? which focuses on organization in latent space is not correlated with the KL term. We present both a vanilla autoencoder without the KL objective and a SketchEmbedNet trained with a KL objective. We observe a drop in overall generation quality in the Conceptual Composition decoding as is expected with an additional constraint but maintained performance in the other tasks. Meanwhile, the autoencoder does not demonstrate any marked improvements over the Conv-VAE in the main paper or any other baseline comparison.

M ADDITIONAL COMPOSITIONALITY MODES

We provide additional clustering methods t-SNE (Maaten & Hinton, 2008) and PCA as well as 2 new experiments that explore the compositionality of our latent SketchEmbedding.

Additional clustering methods We include additional t-SNE and PCA results of the experiments in the main paper. These are presented in Figures 7, 8, 9 10, 11. t-SNE and UMAP are stochastic and do not always produce the same visualization while PCA is deterministic and prioritizes the most important dimensions.

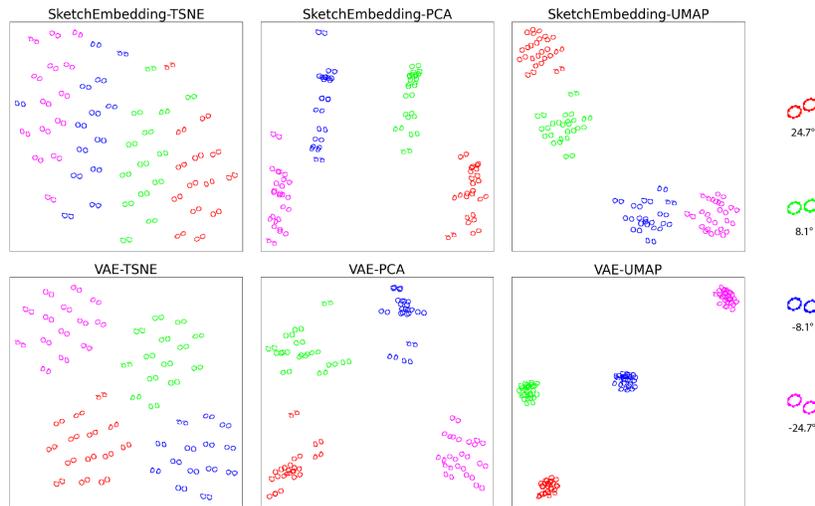


Figure 7: 2D Embedding visualization of different spatial orientations of circles and squares

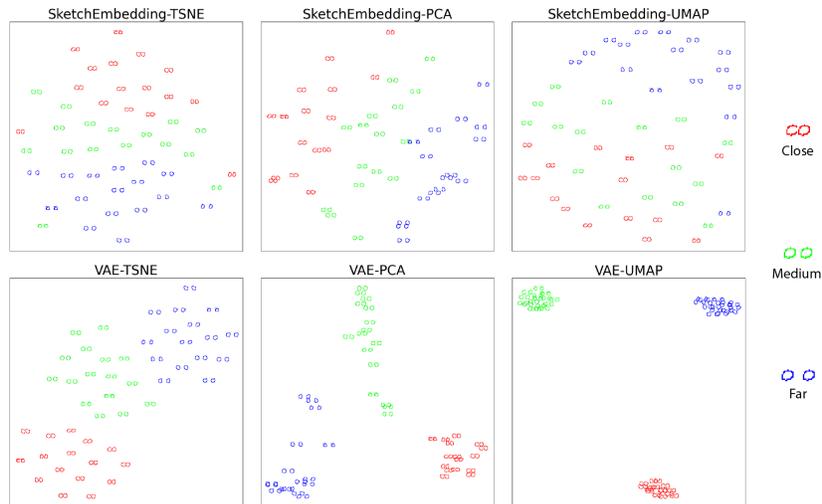


Figure 8: 2D Embedding visualization of different linear distances between shapes

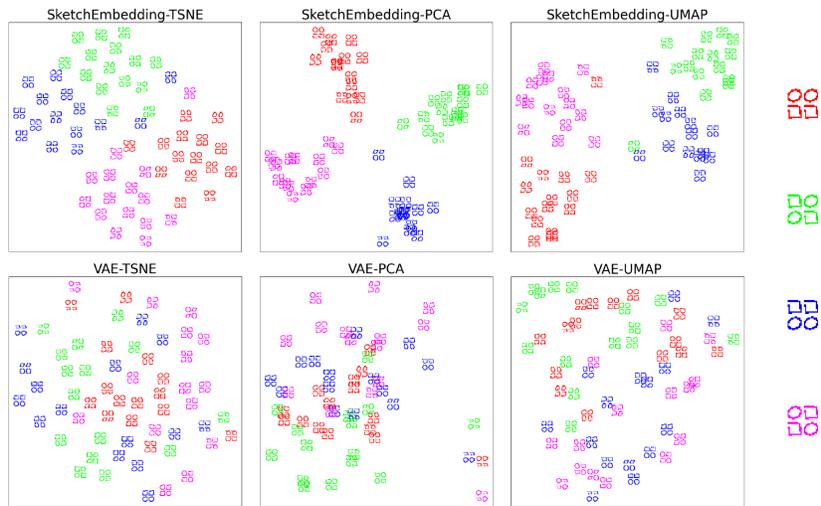


Figure 9: Latent space visualization squares and circles arranged differently in a 2x2 array

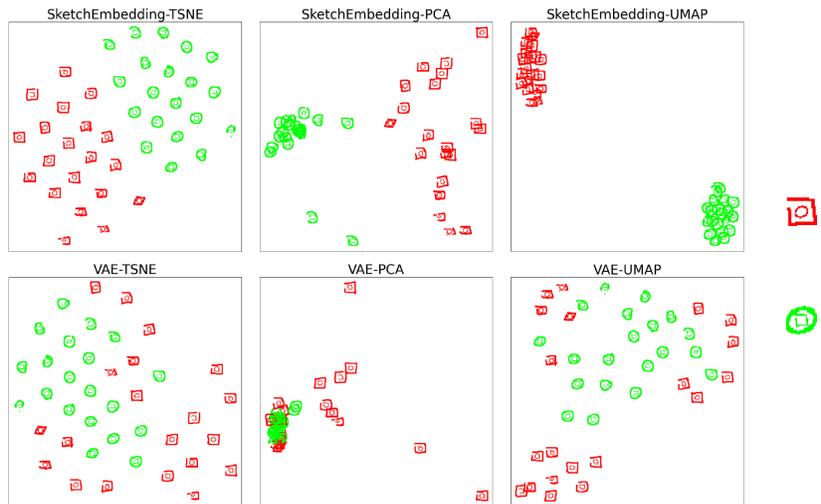


Figure 10: Latent space visualization of composing circles and squares within one another or outside

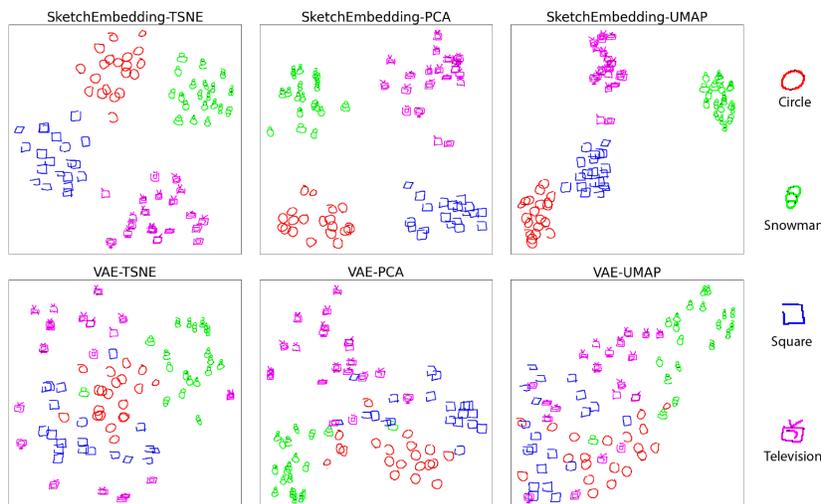


Figure 11: Latent space visualization of composing multiple circles and squares in real sketch drawings

Additional Experiments Here we provide different investigations into the compositionality of our learned embedding space that were not present in our main paper. These results presented in Figure 12 and 13.

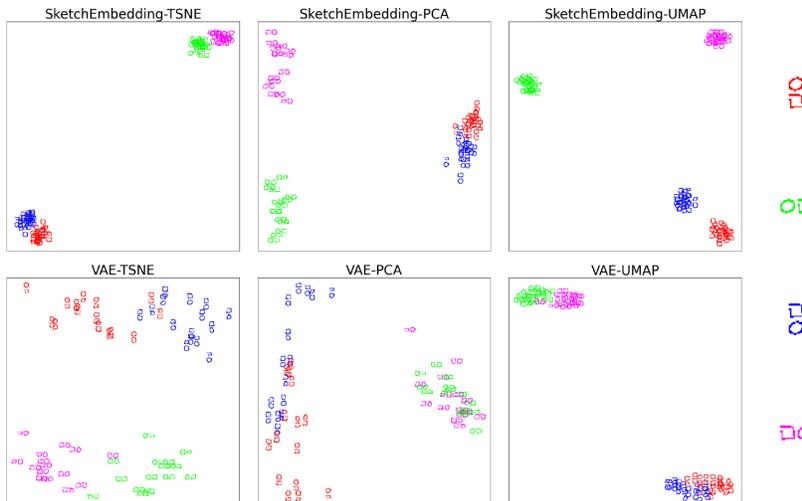


Figure 12: 2D Embedding visualization of different spatial orientations of circles and squares

In Figure 12 we place a square in the center of the example and place a circle above, below or to the sides of it. Once again we find that our SketchEmbedding embedding clusters better than the VAE approach.

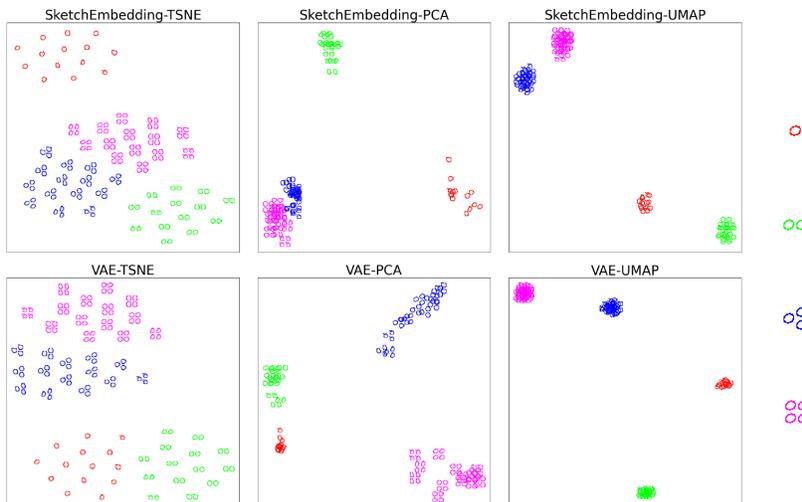


Figure 13: Latent space visualization of composing multiple circles and squares in real sketch drawings

New examples are generated where each class has a different numbers of circles. Both the VAE approach and our SketchEmbedding cluster well and neither appear to learn the count manifold.

N HYPERNETWORK ACTIVATIONS

To further explore how our network understands drawings, we examine the relationships between the activations of the hypernetwork of our HyperLSTM (Ha et al., 2017).

The hypernetwork determines the weights of the LSTM that generates the RNN at each decoding timestep. These activations are 512-dimensional vectors. We collect the activations from many

examples, cluster them in 512-dimensional space and visualize the strokes belonging to each cluster for each example. A full decoding is also rendered where each cluster within an example is assigned a color.

Single class: snowman First we explore this clustering using only the snowman class from Quickdraw (Jongejan et al., 2016). We expect substantial reuse of a "circle" both within and over many examples. Clustering of the strokes is done with the DBSCAN (Ester et al., 1996) and parameter $\epsilon = 3.9$. Results are in Figure 14. Each row is a separate input; the far left column is the color-coded, composed image, the second is the noise cluster and every subsequent column is a unique cluster.

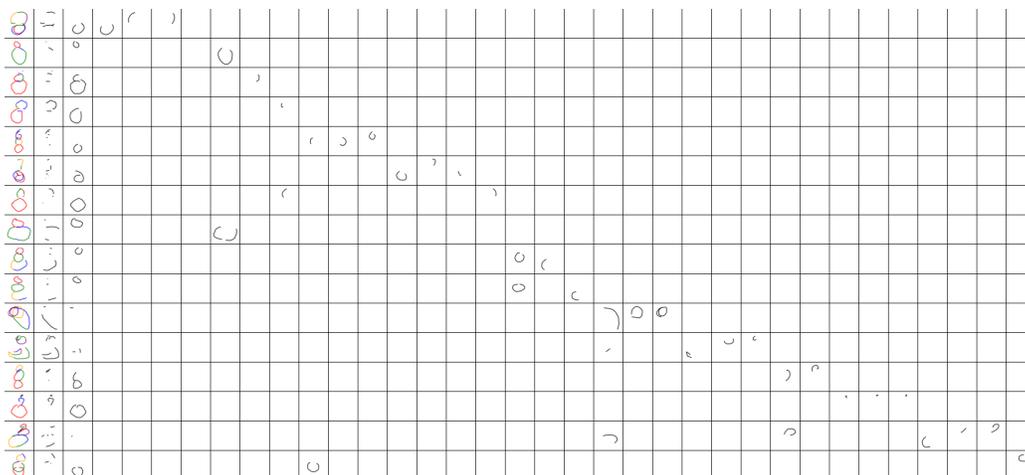


Figure 14: Snowman class stroke clustering

While cluster re-use is limited, cluster 0 often contains a large, fully enclosed circle. Many other clusters may contain circles or partial strokes with some reuse. Larger, fully composed and coloured sketches are presented in Figure 15



Figure 15: Fully composed images with coloured cluster assignments

Many classes: round objects We repeat the above experiment with a mixture of classes that generally can be expected to contain circles. These classes were circles, snowmen, clocks and cups. The two former classes are frequently composed only of circles while the latter are expected to consistently contain other distinct shapes. Results are presented in Figure 16 and select examples in Figure 17.

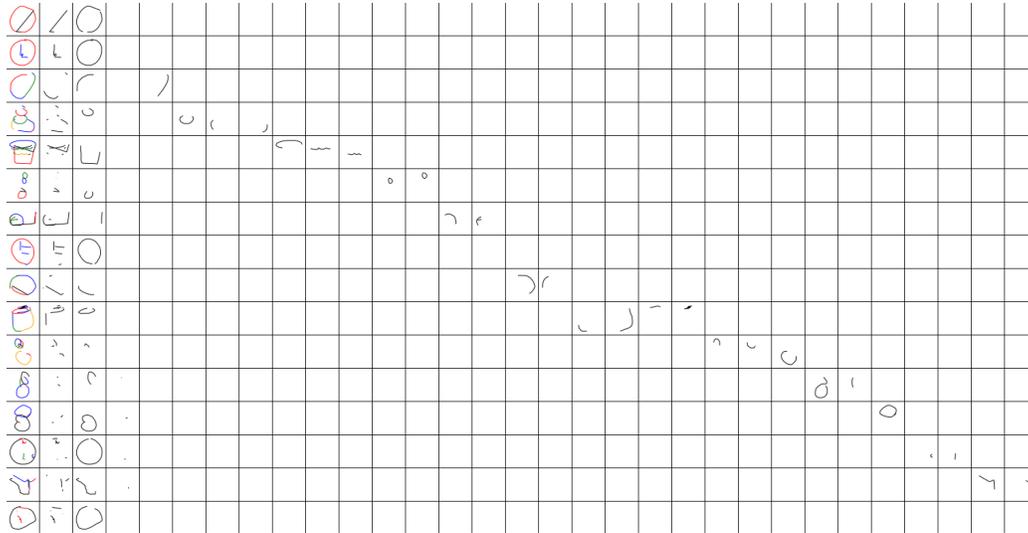


Figure 16: Snowman class stroke clustering

We still observe that the model continues to isolate circles in the first column and note it continues to do so for the cup and clock classes which are not exclusively circular.



Figure 17: Fully composed images with coloured cluster assignments

Many random classes: Finally, we repeat the above clustering with the 45 randomly selected holdout classes from the Quickdraw training process of SketchEmbedding. Results are once again presented in Figure 18 and select examples in Figure 19.

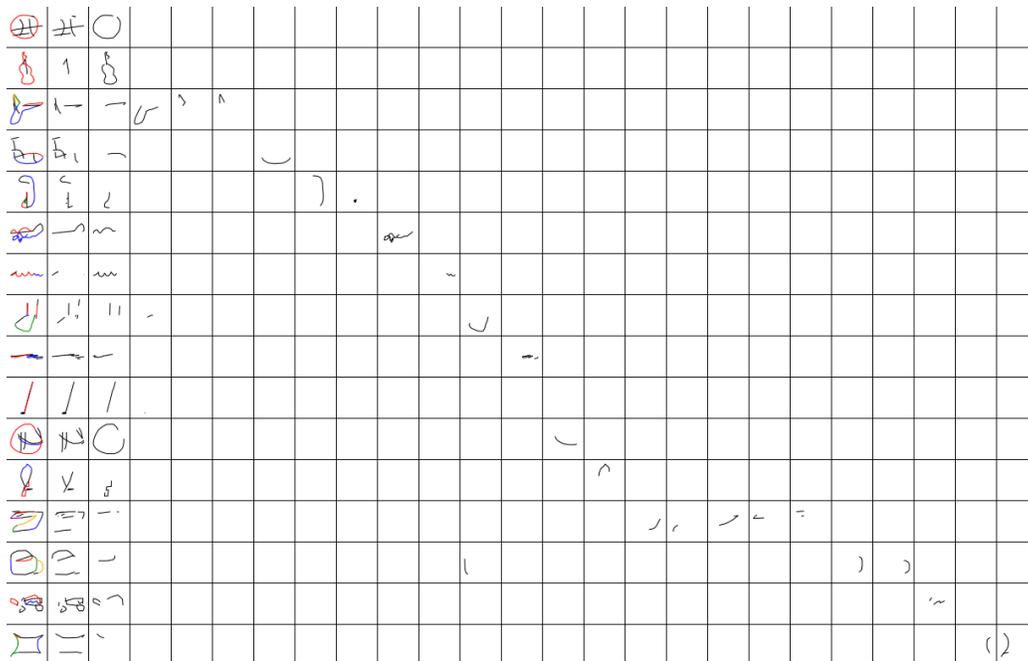


Figure 18: Snowman class stroke clustering



Figure 19: Fully composed images with coloured cluster assignments

REFERENCES

- Antreas Antoniou and Amos J. Storkey. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *CoRR*, abs/1902.09884, 2019.
- Nan Cao, Xin Yan, Yang Shi, and Chaoran Chen. AI-Sketcher: A deep generative model for producing high-quality sketches. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, 2019.
- Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *CoRR*, abs/1709.04121, 2017.
- David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. 1973.
- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD*, 1996.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, 2017.
- Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, Alex Lavin, and D. Scott Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368), 2017. ISSN 0036-8075. doi: 10.1126/science.aag2612.
- Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.
- Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.
- David Ha and Douglas Eck. A neural representation of sketch drawings. In *6th International Conference on Learning Representations, ICLR*, 2018.
- David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR*, 2017.
- Luke B. Hewitt, Maxwell I. Nye, Andreea Gane, Tommi S. Jaakkola, and Joshua B. Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI*, 2018.
- Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL <https://openreview.net/forum?id=Sy2fzU9gl>.
- Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *7th International Conference on Learning Representations, ICLR*, 2019.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
- J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The quick, draw! - A.I. experiment., 2016. URL <https://quickdraw.withgoogle.com/>.
- Siavash Khodadadeh, Ladislau Bölöni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.
- Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050.

- Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. The omniglot challenge: a 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104, Oct 2019.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- M. Liwicki and H. Bunke. Iam-ondb - an on-line english sentence database acquired from handwritten text on a whiteboard. In *Eighth International Conference on Document Analysis and Recognition, ICDAR*, 2005.
- Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- Boris N. Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.
- Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30, NIPS*, 2017.
- Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning to sketch with shortcut cycle consistency. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018.
- Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29, NIPS*, 2016.
- Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M. Hospedales, and Chen Change Loy. Sketch me that shoe. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.