SketchEmbedNet: Learning Novel Concepts by Imitating Drawings

Alexander Wang [*1 2] **Mengye Ren** [*1 2] **Richard S. Zemel** [1 2 3]

Abstract

Sketch drawings capture the salient information of visual concepts. Previous work has shown that neural networks are capable of producing sketches of natural objects drawn from a small number of classes. While earlier approaches focus on generation quality or retrieval, we explore properties of image representations learned by training a model to produce sketches of images. We show that this generative, class-agnostic model produces informative embeddings of images from novel examples, classes, and even novel datasets in a few-shot setting. Additionally, we find that these learned representations exhibit interesting structure and compositionality.

1. Introduction

Drawings are frequently used to facilitate the communication of new ideas. If someone asked what an apple is, or looks like, a natural approach would be to provide a simple, pencil and paper drawing; perhaps a circle with divots on the top and bottom and a small rectangle for a stem. These sketches constitute an intuitive and succinct way to communicate concepts through a prototypical, visual representation. This phenomenon is also preserved in logographic writing systems such as Chinese hanzi and Egyptian hieroglyphs where each character is essentially a sketch of the object it represents. Frequently, humans are able to communicate complex ideas in a few simple strokes.

Inspired by this idea that sketches capture salient aspects of concepts, we hypothesize that it is possible to learn informative representations by expressing them as sketches. In this paper we target the image domain and seek to develop representations of images from which sketch drawings can be generated. Recent research has explored a wide variety of sketch generation models, ranging from generative adversarial networks (GANs) (Isola et al., 2017; Li et al., 2019), to autoregressive (Gregor et al., 2015; Ha & Eck, 2018; Chen et al., 2017), transformer (Ribeiro et al., 2020; Aksan et al., 2020), hierarchical Bayesian (Lake et al., 2015) and neuro-symbolic (Tian et al., 2020) models. These methods may generate in pixel-space or in a sequential setting such as a motor program detailing pen movements over a drawing canvas. Many of them face shortcomings with respect to representation learning on images: hierarchical Bayesian models scale poorly, others only generate a single or a few classes at a time, and many require sequential inputs, which limit their use outside of creative applications.

We develop SketchEmbedNet, a class-agnostic encoder-decoder model that produces a "SketchEmbedding" of an input image as an encoding which is then decoded as a sequential motor program. By knowing "how to sketch an image," it learns an informative representation that leads to strong performance on classification tasks despite being learned without class labels. Additionally, training on a broad collection of classes enables strong generalization and produces a class-agnostic embedding function. We demonstrate these claims by showing that our approach generalizes to novel examples, classes, and datasets, most notably in a challenging unsupervised few-shot classification setting on the Omniglot (Lake et al., 2015) and mini-ImageNet (Vinyals et al., 2016) benchmarks.

While pixel-based methods produce good visual results, they may lack clear component-level awareness, or understanding of the spatial relationships between them in an image; we have seen this collapse of repeated components in GAN literature (Goodfellow, 2017). By incorporating specific pen movements and the contiguous definition of visual components as points through time, SketchEmbeddings encode a unique visual understanding not present in pixel-based methods. We study the presence of componential and spatial understanding in our experiments and also present a surprising phenomenon of conceptual composition where concepts can be added and subtracted through embeddings.

[*]Equal contribution [1]University of Toronto, Toronto, Canada [2]Vector Institute [3]CIFAR. Correspondence to: Alexander Wang <alexw@cs.toronto.edu>, Mengye Ren <mren@cs.toronto.edu>, Richard S. Zemel <zemel@cs.toronto.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

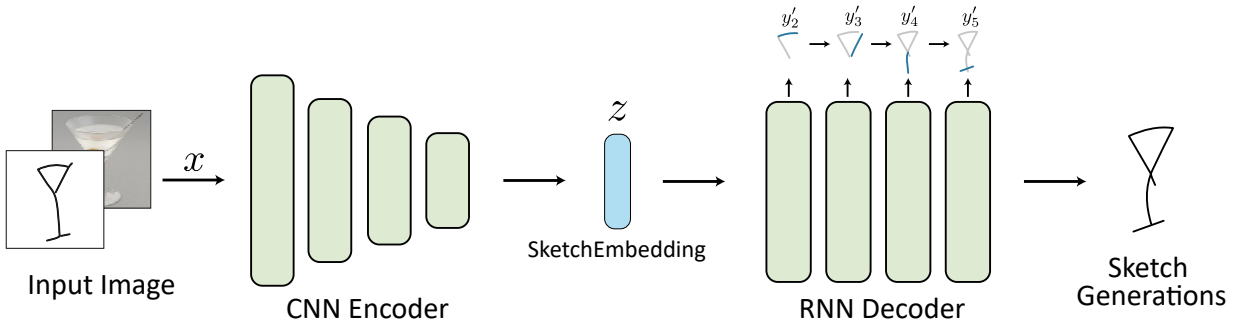


Figure 1: **Representation learning and generation setup** – a pixel image is encoded as SketchEmbedding z then decoded as a sequential sketch. Model input can be either a sketch image or a natural image.

2. Related Work

Sketch-based visual understanding. Recent research motivates the use of sketches to understand and classify images. Work by (Hertzmann, 2020) demonstrated that line drawings are an informative depiction of shape and are intuitive to human perception. Lamb et al. (2020) further, proposed that sketches are a detail-invariant representation of objects in an image that summarize salient visual information. Geirhos et al. (2019) demonstrates that a shape-biased perception, is more robust and reminiscent of human perception. We build on this intuition to sketches for shape-biased perception by building a generative model to capture it in a latent representation.

Sequential models for sketch generation. Many works study the generation of sequential sketches without specifying individual pixel values; Hinton & Nair (2005) trained a generative model for MNIST (LeCun et al., 1998) examples by specifying spring stiffness to move a pen in 2D space. Graves (2013) introduced the use of an LSTM (Hochreiter & Schmidhuber, 1997) to model handwriting as a sequence of points using recurrent networks. SketchRNN (Ha & Eck, 2018) extended the use of RNNs to sketching models that draw a single class. Song et al. (2018); Chen et al. (2017); Ribeiro et al. (2020) made use of pixel inputs and consider more than one class while Ribeiro et al. (2020); Aksan et al. (2020) introduced a transformer (Vaswani et al., 2017) architecture to model sketches. Lake et al. (2015) used a symbolic, hierarchical Bayesian model to generate Omniglot (Lake et al., 2015) examples while Tian et al. (2020) used a neuro-symbolic model for concept abstraction through sketching. Carlier et al. (2020) explored the sequential generation of scalable vector graphics (SVG) images. We leverage the SketchRNN decoder for autoregressive sketch generation, but extend it to hundreds of classes with the focus of learning meaningful image representations. Our model is reminiscent of (Chen et al., 2017) but to our knowledge no existing works have learned a class-agnostic sketching model using pixel image inputs.

Pixel-based drawing models. Sketches and other drawing-like images can be specified directly in pixel space by outputting pixel intensity values. They were proposed as a method to learn general visual representation in the early literature of computer vision (Marr, 1982). Since then pixel-based "sketch images" can be generated through style transfer and low-level processing techniques such as edge detection (Arbelaez et al., 2011). Deep generative models (Isola et al., 2017) using the GAN (Goodfellow et al., 2014) architecture have performed image-sketch domain translation and Photosketch (Li et al., 2019) focused specifically on the task with an $1:N$ image:sketch pairing. Liu et al. (2020) generates sketch images using varying lighting and camera perspectives combined with 3D mesh information. Zhang et al. (2015) used a CNN model to generate sketch-like images of faces. DRAW (Gregor et al., 2015) autoregressively generates sketches in pixel space by using visual attention. van den Oord et al. (2016); Rezende et al. (2016) autoregressively generate pixel drawings. In contrast to pixel-based approaches, SketchEmbedNet does not directly specify pixel intensity and instead produces a sequence of strokes that can be directly rendered into a pixel image. We find that grouping pixels as "strokes" improves the object awareness of our embeddings.

Representation learning using generative models. Frequently, generative models have been used as a method of learning useful representations for downstream tasks of interest. In addition to being one of the first sketch-generation works, Hinton & Nair (2005) also used the inferred motor program to classify MNIST examples without class labels. Many generative models are used for representation learning via an *analysis-by-synthesis* approach, e.g., deep and variational autoencoders (Vincent et al., 2010; Kingma & Welling, 2014), Helmholtz Machines (Dayan et al., 1995), BiGAN (Donahue et al., 2017), etc. Some of these methods

seek to learn better representations by predicting additional properties in a supervised manner. Instead of including these additional tasks alongside pixel-based reconstruction, we generate in the sketch domain to learn our shape-biased representations.

Sketch-based image retrieval (SBIR). SBIR also seeks to map sketches and sketch images to image space. The area is split into fine-grained (FG-SBIR) (Yu et al., 2016; Sangkloy et al., 2016; Bhunia et al., 2020) and a zero-shot setting (ZS-SBIR) (Dutta & Akata, 2019; Pandey et al., 2020; Dey et al., 2019). FG-SBIR considers minute details, while ZS-SBIR learns high-level cross-domain semantics and a joint latent space to perform retrieval.

3. Learning to Imitate Drawings

We present a generative sketching model that outputs a sequential motor program "sketch" describing pen movements, given only an input image. It uses a CNN-encoder and an RNN-decoder trained using our novel pixel-loss curricula in addition to the objectives introduced in SketchRNN (Ha & Eck, 2018).

3.1. Data representation

SketchEmbedNet is trained using image-sketch pairs $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x} \in \mathbb{R}^{H \times W \times C}$ is the input image and $\boldsymbol{y} \in \mathbb{R}^{T \times 5}$ is the motor-program representing a sketch. We adopt the same representation of $\boldsymbol{y}$ as used in SketchRNN (Ha & Eck, 2018). T is the maximum sequence length of the sketch data $\boldsymbol{y}$, and each "stroke" $\boldsymbol{y}_t$ is a pen movement that is described by 5 elements, $(\Delta_x, \Delta_y, s_1, s_2, s_3)$. The first 2 elements are horizontal and vertical displacements on the drawing canvas from the endpoint of the previous stroke. The latter 3 elements are mutually exclusive pen states: s_1 indicates the pen is on paper for the next stroke, s_2 indicates the pen is lifted, and s_3 indicates the sketch sequence has ended. The first "stroke" $\boldsymbol{y}_0$ is initialized as (0, 0, 1, 0, 0) for autoregressive generation. Note that no class information is ever provided to the model while learning to draw.

3.2. Convolutional image embeddings

We use a CNN to encode the input image $\boldsymbol{x}$ and obtain the latent space representation $\boldsymbol{z}$, as shown in Figure 1. To model intra-class variance, $\boldsymbol{z}$ is a Gaussian random variable parameterized by CNN outputs μ and σ like in a VAE (Kingma & Welling, 2014). Throughout this paper, we refer to $\boldsymbol{z}$ as the *SketchEmbedding*.

3.3. Autoregressive decoding of sketches

The RNN decoder used in SketchEmbedNet is the same as in SketchRNN (Ha & Eck, 2018). The decoder outputs a mixture density representing the distribution of the pen offsets at each timestep. It is a mixture of M bivariate Gaussians denoting the spatial offsets as well as the probability over the three pen states s_{1-3}. The spatial offsets $\boldsymbol{\Delta} = (\Delta x, \Delta y)$ are sampled from the M mixture of Gaussians, described by: (1) the normalized mixture weight π_j; (2) mixture means $\boldsymbol{\mu}_j = (\mu_x, \mu_y)_j$; and (3) covariance matrices Σ_j. We further reparameterize each Σ_j with its standard deviation $\boldsymbol{\sigma}_j = (\sigma_x, \sigma_y)_j$ and correlation coefficient $\rho_{xy,j}$. Thus, the stroke offset distribution is

$$p(\boldsymbol{\Delta}) = \sum_{j=1}^{M} \pi_j \mathcal{N}(\boldsymbol{\Delta} | \boldsymbol{\mu}_j, \Sigma_j). \tag{1}$$

The RNN is implemented using a HyperLSTM (Ha et al., 2017); LSTM weights are generated at each timestep by a smaller recurrent "hypernetwork" to improve training stability. Generation is autoregressive, using $\boldsymbol{z} \in \mathbb{R}^D$, concatenated with the stroke from the previous timestep $\boldsymbol{y}_{t-1}$, to form the input to the LSTM. Stroke $\boldsymbol{y}_{t-1}$ is the ground truth supervision at train time (teacher forcing), or a sample $\boldsymbol{y}'_{t-1}$, from the mixture distribution output by the model during from timestep $t-1$.

3.4. Training objectives

We train the drawing model in an end-to-end fashion by jointly optimizing three losses: a pen loss $\mathcal{L}_{\text{pen}}$ for learning pen states, a stroke loss $\mathcal{L}_{\text{stroke}}$ for learning pen offsets, and our proposed pixel loss $\mathcal{L}_{\text{pixel}}$ for matching the visual similarity of the predicted and the target sketch:

$$\mathcal{L} = \mathcal{L}_{\text{pen}} + (1-\alpha)\mathcal{L}_{\text{stroke}} + \alpha\mathcal{L}_{\text{pixel}}, \tag{2}$$

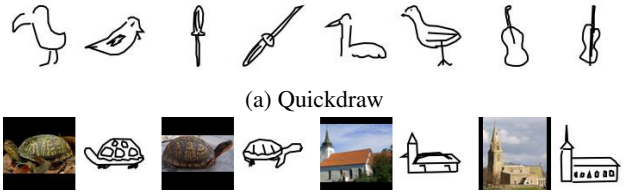
where α is a loss weighting hyperparameter. Both $\mathcal{L}_{\text{pen}}$ and $\mathcal{L}_{\text{stroke}}$ were used in SketchRNN, while the $\mathcal{L}_{\text{pixel}}$ is a novel contribution to stroke-based generative models. Unlike SketchRNN, we do not impose a prior using KL divergence as we are not interested in unconditional sampling, and we found it had a negative impact on the experiments reported below.

Pen loss. The pen-states predictions $\{s'_1, s'_2, s'_3\}$ are optimized as a simple 3-way classification with the softmax cross-entropy loss,

$$\mathcal{L}_{\text{pen}} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{3} s_{m,t} \log(s'_{m,t}). \tag{3}$$

Stroke loss. The stroke loss maximizes the log-likelihood of the spatial offsets of each ground truth stroke $\boldsymbol{\Delta}_t$ given the mixture density distribution p_t at each timestep:

$$\mathcal{L}_{\text{stroke}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_t(\boldsymbol{\Delta}_t). \tag{4}$$



(a) Quickdraw

(b) Sketchy

Figure 2: Samples of Quickdraw and Sketchy data. Sketchy examples are paired sketches and natural images.

Pixel loss. While pixel-level reconstruction objectives are common in generative models (Kingma & Welling, 2014; Vincent et al., 2010; Gregor et al., 2015), they do not exist for sketching models. However, they still represent a meaningful form of generative supervision, promoting visual similarity in the generated result. To enable this loss, we developed a novel rasterization function f_{raster} that produces a pixel image from our stroke parameterization of sketch drawings. f_{raster} transforms the stroke sequence $\boldsymbol{y}$ by viewing it as a set of 2D line segments $(l_0, l_1), (l_1, l_2) \dots (l_{T-1}, l_T)$ where $l_t = \sum_{\tau=0}^{t} \boldsymbol{\Delta}_\tau$. Then, for any arbitrary canvas size we can scale the line segments, compute the distance from every pixel on the canvas to each segment and assign a pixel intensity that is inverse to the shortest distance.

To compute the loss, we apply f_{raster} and a Gaussian blurring filter $g_{\text{blur}}(\cdot)$ to both our prediction $\boldsymbol{y'}$ and ground truth $\boldsymbol{y}$ then compute the binary cross-entropy loss. The Gaussian blur is used to reduce the strictness of our pixel-wise loss.

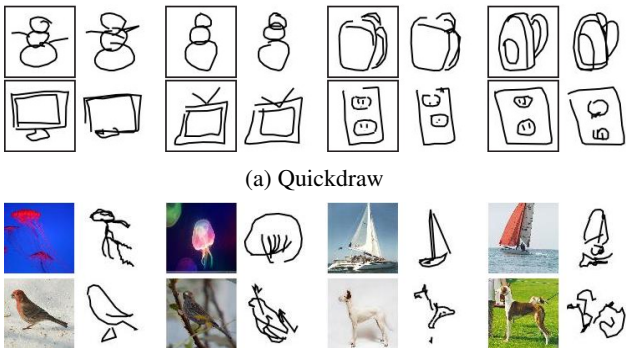
$$I = g_{\text{blur}}(f_{\text{raster}}(\boldsymbol{y})), \quad I' = g_{\text{blur}}(f_{\text{raster}}(\boldsymbol{y'})) \tag{5}$$

$$\mathcal{L}_{\text{pixel}} = -\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} I_{ij} \log(I'_{ij}). \tag{6}$$

Curriculum training schedule. We find that α (in Equation 2) is an important hyperparameter that impacts both the learned embedding space and SketchEmbedNet. A curriculum training schedule is used, increasing α to prioritize $\mathcal{L}_{\text{pixel}}$ relative to $\mathcal{L}_{\text{stroke}}$ as training progresses; this makes intuitive sense as a single drawing can be produced by many stroke sequences but learning to draw in a fixed manner is easier. While $\mathcal{L}_{\text{pen}}$ promotes reproducing a specific drawing sequence, $\mathcal{L}_{\text{pixel}}$ only requires that the generated drawing visually matches the image. Like a human, the model should learn to follow one drawing style (à la paint-by-numbers) before learning to draw freely.

4. Experiments

In this section, we present our experiments on SketchEmbedNet and investigate the properties of SketchEmbeddings.



(a) Quickdraw

(b) Sketchy

Figure 3: Generated sketches of unseen examples from classes seen during training. **Left**–input; **right**–generated image

SketchEmbedNet is trained on diverse examples of sketch–image pairs that do not include any semantic class labels. After training, we freeze the model weights and use the learned CNN encoder as the embedding function to produce SketchEmbeddings for various input images. We study the generalization of SketchEmbeddings through classification tasks involving novel examples, classes and datasets. We then examine emergent spatial and compositional properties of the representation and evaluate model generation quality.

4.1. Training by drawing imitation

We train our drawing model on two different datasets that provide sketch supervision.

- **Quickdraw** (Jongejan et al., 2016) (Figure 2a) pairs sketches with a line drawing "rendering" of the motor program and contains 345 classes of 70,000 examples, produced by human players participating in the game "Quick, Draw!" 300 of 345 classes are randomly selected for training; $\boldsymbol{x}$ is rasterized to a resolution of 28×28 and stroke labels $\boldsymbol{y}$ padded up to length $T = 64$. Any drawing samples exceeding this length were discarded. Data processing procedures and class splits are in Appendix C.

- **Sketchy** (Sangkloy et al., 2016) (Figure 2b) is a more challenging collection of (photorealistic) natural image–sketch pairs and contains 125 classes from ImageNet (Deng et al., 2009), selected for "sketchability". Each class has 100 natural images paired with up to 20 loosely aligned sketches for a total of 75,471 image–sketch pairs. Images are resized to 84×84 and padded to increase spatial agreement; sketch sequences are set to a max length $T = 100$. Classes that overlap with the test set of mini-ImageNet (Ravi & Larochelle, 2017) are removed from our training set, to faithfully

Table 1: Few-shot classification results on Omniglot

Omniglot			(way, shot)			
Algorithm	**Encoder**	**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
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
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
Contrastive	Conv4	Omniglot*	77.69 ± 0.40	92.62 ± 0.20	62.99 ± 0.25	83.70 ± 0.16
SketchEmbedNet *(Ours)*	Conv4	Omniglot*	**94.88** ± 0.22	**99.01** ± 0.08	**86.18** ± 0.18	**96.69** ± 0.07
Contrastive	Conv4	Quickdraw*	83.26 ± 0.40	94.16 ± 0.21	73.01 ± 0.25	86.66 ± 0.17
SketchEmbedNet *(Ours)*	Conv4	Quickdraw*	**96.96** ± 0.17	**99.50** ± 0.06	**91.67** ± 0.14	**98.30** ± 0.05
MAML *(Supervised)* (Finn et al., 2017)	Conv4	Omniglot	94.46 ± 0.35	98.83 ± 0.12	84.60 ± 0.32	96.29 ± 0.13
ProtoNet *(Supervised)* (Snell et al., 2017)	Conv4	Omniglot	98.35 ± 0.22	99.58 ± 0.09	95.31 ± 0.18	98.81 ± 0.07

* Sequential sketch supervision used for training

evaluate few-shot classification performance.

Data samples are presented in Figure 2; for Quickdraw, the input image $\boldsymbol{x}$ and the rendered sketch $\boldsymbol{y}$ are the same. We train a single model on Quickdraw using a 4-layer CNN (Conv4) encoder (Vinyals et al., 2016) and another on the Sketchy dataset with a ResNet-12 (Oreshkin et al., 2018) encoder architecture.

Baselines. We consider the following baselines to compare with SketchEmbedNet.

- **Contrastive** is similar to the search embedding of Ribeiro et al. (2020); a metric learning baseline that matches CNN image embeddings with corresponding RNN sketch embeddings. Our baseline is trained using the InfoNCE loss (van den Oord et al., 2018).
- **Conv-VAE** (Kingma & Welling, 2014) performs pixel-level representation learning without motor program information.
- **Pix2Pix** (Isola et al., 2017) is a generative adversarial approach that performs image to sketch domain transfer but is supervised by sketch images and not the sequential motor program.

Note that Contrastive is an important comparison for SketchEmbedNet as it also uses the motor-program sequence when training on sketch-image pairs.

Implementation details. SketchEmbedNet is trained for 300k iterations with batch size of 256 for Quickdraw and 64 for Sketchy due to memory constraints. Initial learning rate is 1e-3 decaying by 0.85 every 15k steps. We use the Adam (Kingma & Ba, 2015) optimizer and clip gradient values to 1.0. Latent space $\dim(z) = 256$, RNN output size is 1024, and hypernetwork embedding is 64. Mixture count is $M = 30$ and Gaussian blur from $\mathcal{L}_{\text{pixel}}$ uses $\sigma = 2.0$.

Conv4 encoder is identical to Vinyals et al. (2016) and the ResNet-12 encoder uses 4 blocks of 64-128-256-512 filters with ReLU activations. α is set to 0 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. See Appendix B for additional details.

4.2. Few-Shot Classification using SketchEmbeddings

SketchEmbedNet transforms images to strokes, the learned, shape-biased representations could be useful for explaining a novel concept. In this section, we evaluate the ability of learning novel concepts from unseen datasets using few-shot classification benchmarks on Omniglot (Lake et al., 2015) and mini-ImageNet (Vinyals et al., 2016). In few-shot classification, models learn a set of novel classes from only a few examples. We perform few-shot learning on standard N-way, K-shot episodes by training a simple linear classifier on top of SketchEmbeddings.

Typically, the training data of few-shot classification is fully labelled, and the standard approaches learn by utilizing the labelled training data before evaluation on novel test classes (Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017). Unlike these methods, SketchEmbedNet does not use class labels during training. Therefore, we compare our model to unsupervised few-shot learning methods

Table 2: 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
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	-	-
UMTRA (Khodadadeh et al., 2019)	Conv4	mini-ImageNet	39.93	50.73	61.11	67.15
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
Contrastive	ResNet12	Sketchy*	30.56 ± 0.33	39.06 ± 0.33	45.17 ± 0.33	47.84 ± 0.32
SketchEmbedNet *(ours)*	Conv4	Sketchy*	38.61 ± 0.42	53.82 ± 0.41	63.34 ± 0.35	67.22 ± 0.32
SketchEmbedNet *(ours)*	ResNet12	Sketchy*	**40.39** ± 0.44	**57.15** ± 0.38	**67.60** ± 0.33	**71.99** ± 0.3
MAML *(supervised)* (Finn et al., 2017)	Conv4	mini-ImageNet	46.81 ± 0.77	62.13 ± 0.72	71.03 ± 0.69	75.54 ± 0.62
ProtoNet *(supervised)* (Snell et al., 2017)	Conv4	mini-ImageNet	46.56 ± 0.76	62.29 ± 0.71	70.05 ± 0.65	72.04 ± 0.60

* Sequential sketch supervision used for training

Table 3: Effect of α_{max} on few-shot classification accuracy

α_{max}	0.00	0.25	0.50	0.75	0.95	1.00
Omniglot(20,1)	87.17	87.82	**91.67**	90.59	89.77	87.63
mini-ImageNet(5,1)	38.00	38.75	38.11	**39.31**	38.53	37.78

Table 4: Classification accuracy of novel examples and classes

	300-way Training Classes	45-way Unseen Classes
Random CNN	0.85	16.42
Conv-VAE	18.70	53.06
Contrastive	41.58	70.72
SketchEmbedNet	**42.80**	**75.68**

(a) Quickdraw

Embedding model	ILSVRC Top-1	ILSVRC Top-5
Random CNN	1.58	4.78
Conv-VAE	1.13	3.78
Pix2Pix	1.23	4.29
Contrastive	3.95	10.91
SketchEmbedNet	**6.15**	**16.20**

(b) Sketchy.

CACTUs (Hsu et al., 2019), **AAL** (Antoniou & Storkey, 2019) and **UMTRA** (Khodadadeh et al., 2019). CACTUs is a clustering-based method while AAL and UMTRA use data augmentation to approximate supervision for meta-learning (Finn et al., 2017). We also compare to our baselines that use this sketch information: both SketchEmbedNet and **Contrastive** use motor-program sequence supervision, and **Pix2Pix** (Isola et al., 2017) requires natural and sketch image pairings. In addition to these, we provide supervised few-shot learning results using **MAML** (Finn et al., 2017) and **ProtoNet** (Snell et al., 2017) as references.

Omniglot results. The results on Omniglot (Lake et al., 2015) using the split from Vinyals et al. (2016) are reported in Table 1. SketchEmbedNet obtains the highest classification accuracy when training on the Omniglot dataset. The Conv-VAE and as well as the Contrastive model are outperformed by existing unsupervised methods but not by a huge margin.[1] When training on the Quickdraw dataset SketchEmbedNet sees a substantial accuracy increase and exceeds the classification accuracy of the supervised MAML approach. While our model has arguably more supervision information than the unsupervised methods, our performance gains relative to the Contrastive baseline shows that this does not fully explain the results. Furthermore, our method transfers well from Quickdraw to Omniglot without ever seeing a single Omniglot character.

mini-ImageNet results. The results on mini-ImageNet (Vinyals et al., 2016) using the split from (Ravi & Larochelle, 2017) are reported in Table 2. SketchEmbedNet outperforms existing unsupervised few-shot classification approaches. We report results using both Conv4 and ResNet12 backbones; the latter allows more learning capacity for the drawing imitation task, and consistently achieves better performance. Unlike on the Omniglot benchmark, Contrastive and Conv-VAE perform poorly compared to existing methods, whereas SketchEmbedNet scales well to natural images and again outperforms other

[1] We do not include the Pix2Pix baseline here as the input and output images are the same.

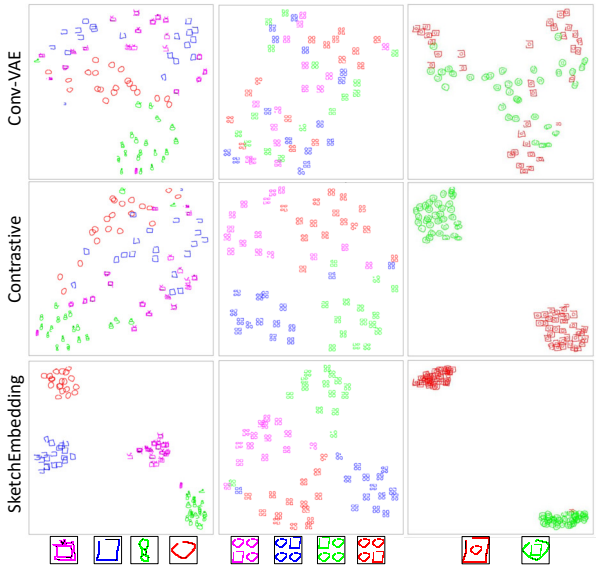


Figure 4: Embedding clustering of images with different component arrangements. **Left**–numerosity; **middle**–placement; **right**–containment

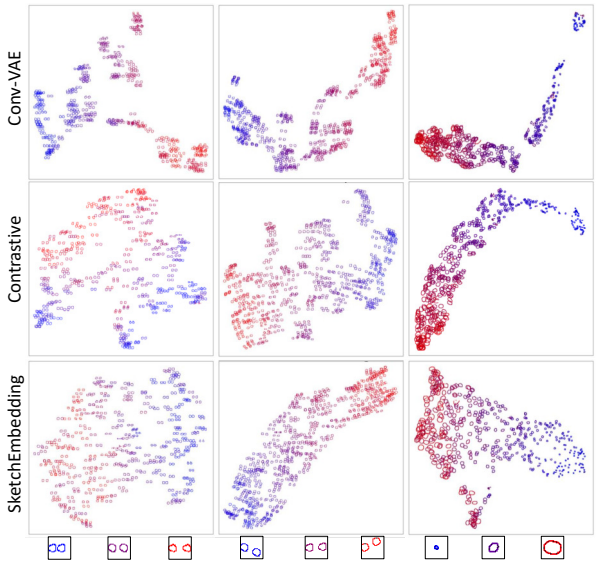


Figure 5: Recovering spatial variables embedded within image components. **Left**–distance; **middle**–angle; **right**–size

unsupervised few-shot learning methods, and even matches the performance of a supervised ProtoNet on 5-way 50-shot (71.99 vs. 72.04). This suggests that forcing the model to generate sketches yields more informative representations.

Effect of pixel-loss weighting. We ablate pixel loss coefficient α_{max} to quantify its impact on the observed representation, using the Omniglot task (Table 3). There is a substantial improvement in few-shot classification when α_{max} is non-zero. α_{max}= 0.50 achieves the best results for Quickdraw, while it trends downwards when α_{max} approaches to 1.0. mini-ImageNet performs best at $\alpha_{\text{max}} = 0.75$ Over-emphasizing the pixel-loss while using teacher forcing causes the model to create sketches by using many strokes, and does not generalize to true autoregressive generation.

4.3. Intra-Dataset Classification

While few-shot classification demonstrates a strong form of generalization to novel classes, and in SketchEmbedNet's case entirely new datasets, we also investigate the useful information learned from the same datasets used in training. Here we study a conventional classification problem: we train a single layer linear classifier on top of input SketchEmbeddings of images drawn from the training dataset. We report accuracy on a validation set of novel images from the same classes, or new classes from the same training dataset.

Quickdraw results. The training data consists of 256 labelled examples for each of the 300 training classes. New example generalization is evaluated in 300-way classification on unseen examples of training classes. Novel class generalization is evaluated on 45-way classification of unseen Quickdraw classes. The results are presented in Table 4a. SketchEmbedNet obtains the best classification performance. The Contrastive method also performs well, demonstrating the informativeness of sketch supervision. Note that while Contrastive performs well on training classes, it performs worse on unseen classes. The few-shot benchmarks in Tables 1, 2 suggest our generative objective is more suitable for novel class generalization. Unlike in the few-shot tasks, a Random CNN performs very poorly likely because the linear classification head lacks the capacity to discriminate the random embeddings.

Sketchy results. Since there are not enough examples or classes to test unseen classes within Sketchy, we evaluate model generalization on 1000-way classification of ImageNet-1K (ILSVRC2012), and the validation accuracy is presented in Table 4b. It is important to note that all the methods shown here only have access to a maximum of 125 Sketchy classes during training, resized down to 84×84, with a max of 100 unique photos per class, and thus they are not directly comparable to current state-of-the-art methods trained on ImageNet. SketchEmbedNet once again obtains the best performance, not only relative to the image-based baselines, Random CNN, Conv-VAE and Pix2Pix, but also to the Contrastive learning model, which like SketchEmbedNet utilizes the sketch information during training. While Contrastive is competitive in Quickdraw classification, it does not maintain this performance on more difficult tasks with natural images, much like in the few-shot natural image setting. Unlike in Quickdraw classification where pretrain-

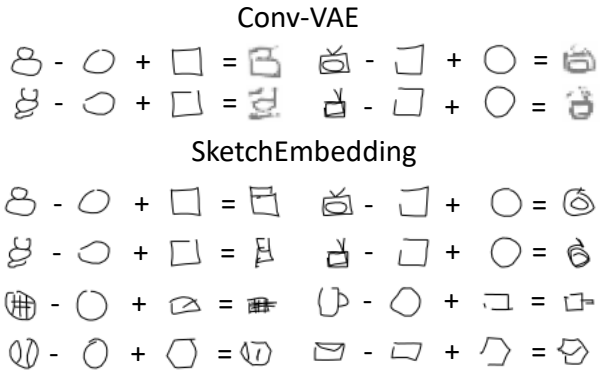


Figure 6: Conceptual composition of image representations. Several sketches are shown for the two models following algebraic operations on their embedding vectors.

ing is effective, all 3 pixel-based methods perform similarly poorly.

4.4. Emergent properties of SketchEmbeddings

Here we probe properties of the image representations formed by SketchEmbedNet and the baseline models. We construct a set of experiments to showcase the spatial and component-level visual understanding and conceptual composition in the embedding space.

Arrangement of image components. To test component-level awareness, we construct image examples containing different arrangements of multiple objects in image space. We then embed these examples and project into 2D space using UMAP (McInnes et al., 2018) to visualize their organization. The leftmost panel of Figure 4 exhibits a numerosity relation with Quickdraw classes containing duplicated components; snowmen with circles and televisions with squares. The next two panels of Figure 4 contain examples with a placement and containment relation. SketchEmbedding representations are the most distinguishable and are easily separable. The pixel-based Conv-VAE is the least distinguishable, while the Contrastive model performs well in the containment case but poorly in the other two. As these image components are drawn contiguous through time and separated by lifted pen states, SketchEmbedNet learns to group the input pixels together as abstract elements to be drawn together.

Recovering spatial relationships. We examine how the underlying variables of distance, angle or size are captured by the studied embedding functions. We construct and embed examples changing each of the variables of interest. The embeddings are again projected into 2D by the UMAP (McInnes et al., 2018) algorithm in Figure 5. After projection, SketchEmbedNet recovers the variable of interest as an approximately linear manifold in 2D space; the Contrastive embedding produces similar results, while the pixel-based Conv-VAE is more clustered and non-linear. This shows that relating images to sketch motor programs encourages the system to learn the spatial relationships between components, since it needs to produce the Δx and Δy values to satisfy the training objective.

Figure 7: Generated sketches of images from datasets unseen during training. **Left**–input; **right**–generated image

	Seen	Unseen
Original Data	97.66	96.09
Conv-VAE	76.28 $\pm$ 0.93	75.07 $\pm$ 0.84
SketchEmbedNet	**81.44** $\pm$ 0.95	**77.94** $\pm$ 1.07

Table 5: Classification accuracy for generated sketch images.

Conceptual composition. Finally, we explore the use of SketchEmbeddings for composing embedded concepts. In natural language literature, vector algebra such as "king" - "man" + "woman" = "queen" (Mikolov et al., 2013) shows linear compositionality in the concept space of word embedding. It has also been demonstrated in human face images and vector graphics (Bojanowski et al., 2018; Shen et al., 2020; Carlier et al., 2020). Here we try to explore such concept compositionality property in sketch image understanding as well. We embed examples of simple shapes such as a square or circle as well as more complex examples like a snowman or mail envelope and perform arithmetic in the latent space. Surprisingly, upon decoding the SketchEmbedding vectors we recover intuitive sketch generations. For example, if we subtract the embedding of a circle from snowman and add a square, then the resultant vector gets decoded into an image of a stack of boxes. We present examples in Figure 6. By contrast, the Conv-VAE does not produce sensible decodings on this task.

4.5. Evaluating generation quality

Another method to evaluate our learned image representations is through the sketches generated based on these representations; a good representation should produce a recognizable image. Figures 3 and 7 show that SketchEmbedNet can generate reasonable sketches of training classes as well as unseen data domains. When drawing natural im-

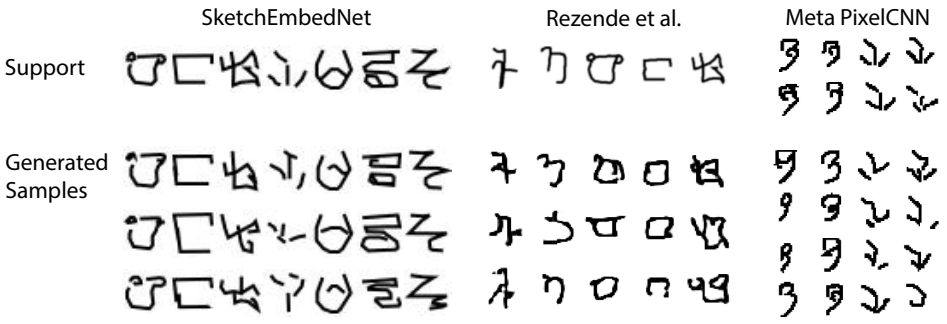


Figure 8: One-shot Omniglot generation compared to Rezende et al. (2016); Reed et al. (2018).

ages, it sketches the general shape of the subject rather than replicating specific details.

Classifying generated examples. Quantative assessment of generated images is often challenging and per-pixel metrics like in (Reed et al., 2018; Rezende et al., 2016) may penalize generative variation that still preserves meaning. We train ResNet classifiers for an Inception Score (Salimans et al., 2016) inspired metric. One classifier is trained on 45 ("seen") Quickdraw training classes and the other on 45 held out ("unseen") classes that were not encountered during model training. Samples generated by a sketching model are rendered, then classified; we report each classifier's accuracy on these examples compared to its training accuracy in Table 5. SketchEmbedNet produces more recognizable sketches than a Conv-VAE model when generating examples of both seen and unseen object classes.

Qualitative comparison of generations. In addition to the Inception-score (Salimans et al., 2016) inspired metric, we also qualitatively assess the generations of SketchEmbedNet on unseen datasets. One-shot generations are sampled from Omniglot (Lake et al., 2015) and are visually compared with other few- and one-shot generation methods (Rezende et al., 2016; Reed et al., 2018) (Figure 8).

None of the models have seen any examples from the character class or parent alphabet. Furthermore, SketchEmbedNet was not trained on any Omniglot data. Visually, our generated images better resemble the support examples and have generative variance that better preserves class semantics. Generations in pixel space may disrupt strokes and alter the character to human perception. This is especially true for written characters as they are frequently defined by a specific set of strokes instead of blurry clusters of pixels.

Discussion. While having a generative objective is useful for representation learning (we see that SketchEmbedNet outperform our Contrastive representations), it is insufficient to guarantee an informative embedding for other tasks. The Conv-VAE generations perform slightly worse on the recognizability task in Table 5, while being significantly worse in our previous classification tasks in Tables 1, 2 and 4.

This suggests that the output domain has an impact on the learned representation. The increased componential and spatial awareness from generating sketches (as in Section 4.4) makes SketchEmbeddings better for downstream classification tasks by better capturing the visual shape in images.

5. Conclusion

Learning to draw is not only an artistic pursuit but drives a distillation of real-world visual concepts. In this paper, we present a model that learns representation of images which capture salient features, by producing sketches of image inputs. While sketch data may be challenging to source, we show that SketchEmbedNet can generalize to image domains beyond the training data. Finally, SketchEmbedNet achieves competitive performance on few-shot learning of novel classes, and represents compositional properties, suggesting that learning to draw can be a promising avenue for learning general visual representations.

Acknowledgments We thank Jake Snell, James Lucas and Robert Adragna for their helpful feedback on earlier drafts of the manuscript. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute (`www.vectorinstitute.ai/#partners`). This project is supported by NSERC and the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements,

either expressed or implied, of IARPA, DoI/IBC, or the U.S.
Government.

References

Aksan, E., Deselaers, T., Tagliasacchi, A., and Hilliges, O. Cose: Compositional stroke embeddings. *Advances in Neural Information Processing Systems*, 33, 2020.

Antoniou, A. and Storkey, A. J. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *CoRR*, abs/1902.09884, 2019.

Arbelaez, P., Maire, M., Fowlkes, C. C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, 2011.

Bhunia, A. K., Yang, Y., Hospedales, T. M., Xiang, T., and Song, Y.-Z. Sketch less for more: On-the-fly fine-grained sketch-based image retrieval. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2020.

Bojanowski, P., Joulin, A., Lopez-Paz, D., and Szlam, A. Optimizing the latent space of generative networks. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

Carlier, A., Danelljan, M., Alahi, A., and Timofte, R. Deepsvg: A hierarchical generative network for vector graphics animation. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33, NeurIPS*, 2020.

Chen, Y., Tu, S., Yi, Y., and Xu, L. Sketch-pix2seq: a model to generate sketches of multiple categories. *CoRR*, abs/1709.04121, 2017.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR*, 2009.

Dey, S., Riba, P., Dutta, A., Llados, J., and Song, Y.-Z. Doodle to search: Practical zero-shot sketch-based image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. In *5th International Conference on Learning Representations, ICLR*, 2017.

Douglas, D. H. and Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. 1973.

Dutta, A. and Akata, Z. Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, 2017.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *7th International Conference on Learning Representations, ICLR*, 2019.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., Lou, X., Meng, Z., Liu, Y., Wang, H., Lavin, A., and Phoenix, D. S. 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.

Goodfellow, I. J. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27, NIPS*, 2014.

Graves, A. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. DRAW: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

Ha, D. and Eck, D. A neural representation of sketch drawings. In *6th International Conference on Learning Representations, ICLR*, 2018.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *5th International Conference on Learning Representations, ICLR*, 2017.

Hertzmann, A. Why do line drawings work? a realism hypothesis. *Perception*, 49:439 – 451, 2020.

Hewitt, L. B., Nye, M. I., Gane, A., Jaakkola, T. S., and Tenenbaum, J. B. 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.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained

variational framework. In *5th International Conference on Learning Representations, ICLR*, 2017.

Hinton, G. E. and Nair, V. Inferring motor programs from images of handwritten digits. In *Advances in Neural Information Processing Systems 18, NIPS*, 2005.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. In *7th International Conference on Learning Representations, ICLR*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

Isola, P., Zhu, J., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2017.

Jongejan, J., Rowley, H., Kawashima, T., Kim, J., and Fox-Gieg., N. The quick, draw! - A.I. experiment., 2016. URL `https://quickdraw.withgoogle.com/`.

Khodadadeh, S., Bölöni, L., and Shah, M. Unsupervised meta-learning for few-shot image classification. In *Advances in Neural Information Processing Systems 32, NeurIPS*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR*, 2014.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. The omniglot challenge: a 3-year progress report. *Current Opinion in Behavioral Sciences*, 29:97–104, Oct 2019.

Lamb, A., Ozair, S., Verma, V., and Ha, D. Sketchtransfer: A new dataset for exploring detail-invariance and the abstractions learned by deep networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, WACV*, 2020.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, M., Lin, Z. L., Mech, R., Yumer, E., and Ramanan, D. Photo-sketching: Inferring contour drawings from images. In *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2019.

Liu, D., Nabail, M., Hertzmann, A., and Kalogerakis, E. Neural contours: Learning to draw lines from 3d shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Marr, D. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982. ISBN 0716715678.

McInnes, L., Healy, J., Saul, N., and Großberger, L. UMAP: uniform manifold approximation and projection. *J. Open Source Softw.*, 3(29):861, 2018.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y. (eds.), *1st International Conference on Learning Representations, ICLR*, 2013.

Oreshkin, B. N., López, P. R., and Lacoste, A. TADAM: task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems 31, NeurIPS*, 2018.

Pandey, A., Mishra, A., Verma, V. K., Mittal, A., and Murthy, H. A. Stacked adversarial network for zero-shot sketch based image retrieval. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision, WACV*, 2020.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR*, 2017.

Reed, S. E., Chen, Y., Paine, T., van den Oord, A., Eslami, S. M. A., Rezende, D. J., Vinyals, O., and de Freitas, N. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *6th International Conference on Learning Representations, ICLR*, 2018.

Rezende, D. J., Mohamed, S., Danihelka, I., Gregor, K., and Wierstra, D. One-shot generalization in deep generative models. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016.

Ribeiro, L. S. F., Bui, T., Collomosse, J., and Ponti, M. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2020.

Salimans, T., Goodfellow, I. J., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29, NIPS*, 2016.

Sangkloy, P., Burnell, N., Ham, C., and Hays, J. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4):119:1–119:12, 2016.

Shen, Y., Gu, J., Tang, X., and Zhou, B. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9243–9252, 2020.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30, NIPS*, 2017.

Song, J., Pang, K., Song, Y., Xiang, T., and Hospedales, T. M. Learning to sketch with shortcut cycle consistency. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2018.

Tian, L., Ellis, K., Kryven, M., and Tenenbaum, J. Learning abstract structure for drawing by efficient motor program induction. *Advances in Neural Information Processing Systems*, 33, 2020.

van den Oord, A., Kalchbrenner, N., Espeholt, L., Kavukcuoglu, K., Vinyals, O., and Graves, A. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems 29, NIPS*, 2016.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30, NIPS*, 2017.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29, NIPS*, 2016.

Yu, Q., Liu, F., Song, Y., Xiang, T., Hospedales, T. M., and Loy, C. C. Sketch me that shoe. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

Zhang, L., Lin, L., Wu, X., Ding, S., and Zhang, L. End-to-end photo-sketch generation via fully convolutional representation learning. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, ICMR*, 2015.

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.

One purpose of the 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:

$$\begin{bmatrix}\Delta x_t \\ \Delta y_t\end{bmatrix} = \begin{bmatrix}\mu_{x,t} \\ \mu_{y,t}\end{bmatrix} + \begin{bmatrix}\sigma_{x,t} & 0 \\ \rho_{xy,t}\sigma_{y,t} & \sigma_{y,t}\sqrt{1-\rho_{xy,t}^2}\end{bmatrix}\boldsymbol{\epsilon} \quad (7)$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}_2). \quad (8)$$

After sampling we compute the cumulative sum of every stroke over the time so that we obtain an absolute position at each timestep:

$$\begin{bmatrix}x_t \\ y_t\end{bmatrix} = \sum_{\tau=0}^{T} \begin{bmatrix}\Delta x_\tau \\ \Delta y_\tau\end{bmatrix}. \quad (9)$$

$$\boldsymbol{y}_{t,abs} = (x_t, y_t, s_1, s_2, s_3). \quad (10)$$

Sketch 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 $\boldsymbol{y}$ and apply them to both the prediction $\boldsymbol{y}'$ as well as the ground truth $\boldsymbol{y}$. These parameters are computed as:

$$\lambda = \min\left\{\frac{W}{x_{\text{max}} - x_{\text{min}}}, \frac{H}{y_{\text{max}} - y_{\text{min}}}\right\}, \quad (11)$$

$$x_{\text{shift}} = \frac{x_{\text{max}} + x_{\text{min}}}{2}\lambda, \;\; y_{\text{shift}} = \frac{y_{\text{max}} + y_{\text{min}}}{2}\lambda. \quad (12)$$

$x_{\text{max}}, x_{\text{min}}, y_{\text{max}}, y_{\text{min}}$ 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.

$$p_{ij} = \sigma\Bigg[2 - 5 \times \min_{t=1\ldots N_s}\Bigg(\quad (13)$$

$$\text{dist}\big((i,j),(x_{t-1},y_{t-1}),(x_t,y_t)\big) + (1 - \lfloor s_{1,t-1} \rceil)10^6\Bigg)\Bigg], \quad (14)$$

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. Data Processing

C.1. Quickdraw

We apply the same data processing methods as in Ha & Eck (2018) with no additional changes to produce our stroke labels $\boldsymbol{y}$. When rasterizing for our input $\boldsymbol{x}$, 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

C.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 $\boldsymbol{y}$. We also rasterize our images in the same manner as above for our input $\boldsymbol{x}$.

C.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.

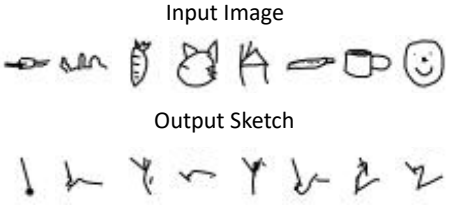
The following classes were use for training after removing overlapping classes with mini-ImageNet: hot-air_balloon, violin, tiger, eyeglasses, mouse, jack-o-lantern, lobster, teddy_bear, teapot, helicopter, duck, wading_bird, rabbit, penguin, sheep, windmill, piano, jellyfish, table, fan, beetle, cabin, 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

D. Pixel-loss Weighting $\alpha_{\max}$ Ablation for Generation Quality

Table 6: Effect of $\alpha_{\max}$ on classification accuracy of generated sketches.

$\alpha_{\max}$	0.00	0.25	0.50	0.75	0.95	1.00
Seen	87.76	87.35	81.44	66.80	36.98	04.80
Unseen	84.02	85.32	77.94	63.10	32.94	04.50

(a) Autoregressive generation.



(b) Teacher-forced generation.

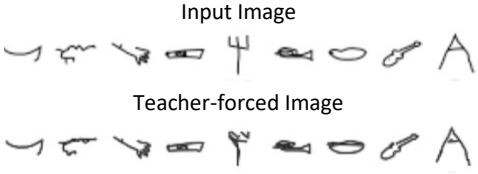


Figure 9: Sketches of SketchEmbedNet trained with $\alpha_{\max} = 1.0$.

We also ablate the impact of pixel-loss weighting parameter $\alpha_{\max}$ on the classification accuracy of the ResNet models from Section 4.5. The evaluation process is the same, generating sketches of examples from classes that were either seen during training or new to the model and classifying them in 45-way classification. Results are shown in Table 6.

Results are only shown for the Quickdraw (Jongejan et al., 2016) setting. Increasing pixel-loss weighting has a minor impact on classification accuracy at lower values but has

a significant detriment at higher weightings. This is due to the teacher-forcing training process. As we de-weight the stroke loss, the model no longer learns to handle the uncertainty of the input position in the space of the 2D canvas by predicting a distribution that explains the next ground truth point. It only matches the generation in pixel space and no longer generates a sensible stroke trajectory on the canvas. While training under teacher forcing, this is not an issue as it is fed the ground truth input point every time, but in autoregressive this generation quickly degrades as each step no longer produces the a point that is a meaningful input for the next time step. We can see the significant difference between generation quality under techer forcing and autoregressive generation in Figure 9.

E. 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 10a and 10b.

We observe that compositionality is also present in these interpolations. In the top row of Figure 10a, 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.

F. 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-based approach against other few-shot classification models on the benchmark. Results are shown in Table 7.

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

G. Effect of Random Seeding on Few-Shot Classification

The training objective for SketchEmbedNetis 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.

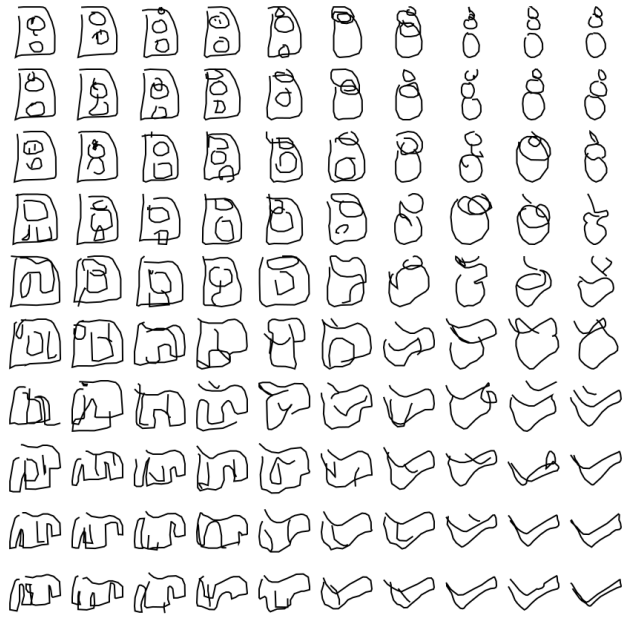
We disregard the per (evaluation) episode variance of our model in each test stage and only present the mean accuracy. We then compute a new confidence interval over random seeds. Results are presented in Tables 8a, 8b.

H. Few-shot Classification on Omniglot – Full Results.

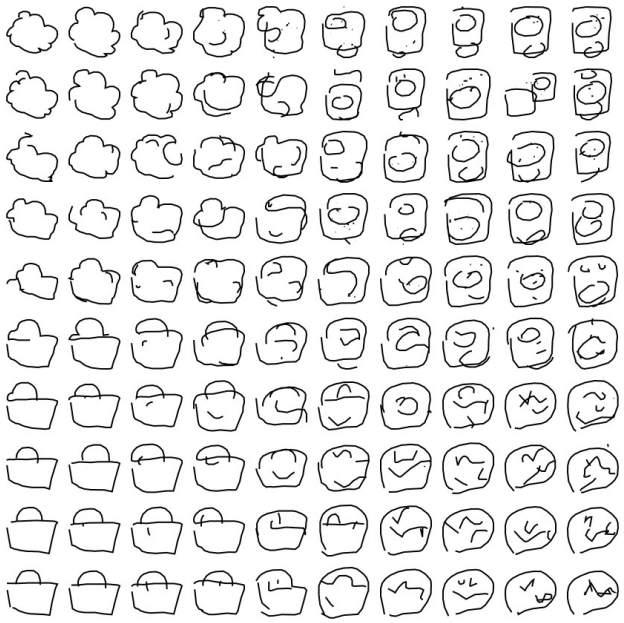
The full results (Table 9) for few-shot classification on the Omniglot (Lake et al., 2015) dataset, including the ResNet12 (Oreshkin et al., 2018) model. We provide results on SketchEmbedNet trained with a KL objective on the latent representation. The (*w/ Labels*) is a model variant where there is an additional head predicting the class from the latent representation while sketching the class. This was to hopefully learn a more discriminative embedding, except it lowered classification accuracy.

I. Few-shot Classification on mini-ImageNet – Full Results

The full results (Table 10) for few-shot classification on the mini-ImageNet dataset, including the ResNet12 (Oreshkin et al., 2018) model and Conv4 models.



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



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

Figure 10: Latent space interpolations of randomly selected examples.

Table 7: Few-shot classification results on Omniglot (Lake split).

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

Table 8: Few-shot classification random seeding experiments.

(a) Omniglot (Conv4).

	(way, shot)			
Seed	**(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 ± 0.12	99.43 ± 0.02	90.69 ± 0.20	98.07 ± 0.05

(b) 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

Table 9: Full table of 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
Contrastive	Conv4	Omniglot*	77.69 ± 0.40	92.62 ± 0.20	62.99 ± 0.25	83.70 ± 0.16
SketchEmbedNet *(Ours)*	Conv4	Omniglot*	94.88 ± 0.22	99.01 ± 0.08	86.18 ± 0.18	96.69 ± 0.07
Contrastive	Conv4	Quickdraw*	83.26 ± 0.40	94.16 ± 0.21	73.01 ± 0.25	86.66 ± 0.17
SketchEmbedNet-avg *(Ours)*	Conv4	Quickdraw*	96.37	99.43	90.69	98.07
SketchEmbedNet-best *(Ours)*	Conv4	Quickdraw*	**96.96** ± 0.17	**99.50** ± 0.06	**91.67** ± 0.14	98.30 ± 0.05
SketchEmbedNet-avg *(Ours)*	ResNet12	Quickdraw*	96.00	99.51	89.88	98.27
SketchEmbedNet-best *(Ours)*	ResNet12	Quickdraw*	96.61 ± 0.19	**99.58** ± 0.06	91.25 ± 0.15	**98.58** ± 0.05
SketchEmbedNet(KL)-avg *(Ours)*	Conv4	Quickdraw*	96.06	99.40	89.83	97.92
SketchEmbedNet(KL)-best *(Ours)*	Conv4	Quickdraw*	96.60 ± 0.18	99.46 ± 0.06	90.84 ± 0.15	98.09 ± 0.06
SketchEmbedNet *(w/ Labels) (Ours)*	Conv4	Quickdraw*	88.52 ± 0.34	96.73 ± 0.13	71.35 ± 0.24	88.16 ± 0.14
MAML *(Supervised)* (Finn et al., 2017)	Conv4	Omniglot	94.46 ± 0.35	98.83 ± 0.12	84.60 ± 0.32	96.29 ± 0.13
ProtoNet *(Supervised)* (Snell et al., 2017)	Conv4	Omniglot	98.35 ± 0.22	99.58 ± 0.09	95.31 ± 0.18	98.81 ± 0.07

* Sequential sketch supervision used for training

Table 10: Full table of 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
Contrastive	ResNet12	Sketchy*	30.56 ± 0.33	39.06 ± 0.33	45.17 ± 0.33	47.84 ± 0.32
SketchEmbedNet-avg *(ours)*	Conv4	Sketchy*	37.01	51.49	61.41	65.75
SketchEmbedNet-best *(ours)*	Conv4	Sketchy*	38.61 ± 0.42	53.82 ± 0.41	63.34 ± 0.35	67.22 ± 0.32
SketchEmbedNet-avg *(ours)*	ResNet12	Sketchy*	38.55	54.39	65.14	69.70
SketchEmbedNet-best *(ours)*	ResNet12	Sketchy*	**40.39** ± 0.44	**57.15** ± 0.38	**67.60** ± 0.33	**71.99** ± 0.3
MAML *(supervised)* (Finn et al., 2017)	Conv4	mini-ImageNet	46.81 ± 0.77	62.13 ± 0.72	71.03 ± 0.69	75.54 ± 0.62
ProtoNet *(supervised)* (Snell et al., 2017)	Conv4	mini-ImageNet	46.56 ± 0.76	62.29 ± 0.71	70.05 ± 0.65	72.04 ± 0.60

* Sequential sketch supervision used for training