Diffusion Models as Artists: Are we Closing the Gap between Humans and Machines?

Victor Boutin [1 2] Thomas Fel [1 2] Lakshya Singhal [2] Rishav Mukherji [2] Akash Nagaraj [2] Julien Colin [2 3] Thomas Serre [1 2]

Abstract

An important milestone for AI is the development of algorithms that can produce drawings that are indistinguishable from those of humans. Here, we adapt the "diversity vs. recognizability" scoring framework from Boutin et al., 2022 and find that one-shot diffusion models have indeed started to close the gap between humans and machines. However, using a finer-grained measure of the originality of individual samples, we show that strengthening the guidance of diffusion models helps improve the humanness of their drawings, but they still fall short of approximating the originality and recognizability of human drawings. Comparing human category diagnostic features, collected through an online psychophysics experiment, against those derived from diffusion models reveals that humans rely on fewer and more localized features. Overall, our study suggests that diffusion models have significantly helped improve the quality of machine-generated drawings; however, a gap between humans and machines remains – in part explainable by discrepancies in visual strategies.

1. Introduction

Drawing is a fundamental human skill; from paintings on cave walls to the finest pieces of art, generations of humans have expressed their creative skills and imagination through drawings (Donald, 1991). Drawings are so deeply rooted in human cognition that they are routinely used in a variety of clinical settings – from evaluating emotional trauma (Koppitz, 1968) and developmental disorders (Ryan-Wenger, 2001) to intellectual deficits (Goodenough, 1926). Cognitive psychologists and computer scientists have also used drawing tasks to probe the human ability to learn new visual concepts from just a single example (Feldman, 1997; Lake et al., 2015). From a computational perspective, such one-shot drawing tasks offer an unprecedented challenge – to solve the seemingly impossible task of estimating entire probability distributions for novel image categories from a unique sample. Nevertheless, humans can effortlessly produce drawings that are original (i.e., sufficiently different from the shown exemplar), yet, easily recognizable (Tiedemann et al., 2022) – suggesting strong inductive biases (Tenenbaum, 1999; Ullman & Tenenbaum, 2020) that are yet to be discovered.

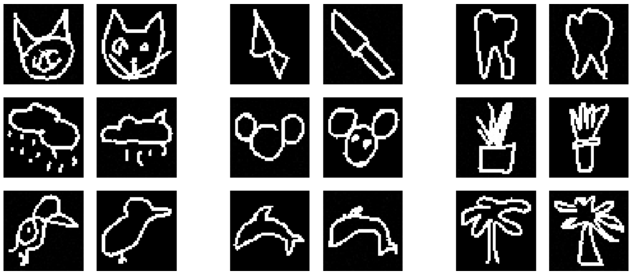


Figure 1. Can you tell apart human from machine-generated samples in each pair[1]? Machine samples are generated using a CFGDM (see Section 2.2).

While a common criticism of modern AI approaches is their appetite for large training datasets, significant progress has been made in the field of few-shot image generation. In particular, one-shot generative models based on Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs) have started to take advantage of various inductive biases via specific forms of spatial attention (Rezende et al., 2016) or the integration of contextual information (Edwards & Storkey, 2016; Giannone & Winther, 2021; Antoniou et al., 2017). Furthermore, the recent breakthrough achieved using diffusion models (Song & Ermon, 2019; Sohl-Dickstein et al., 2015) makes them a par-

[1] Artificial and Natural Intelligence Toulouse Institute, Université de Toulouse, Toulouse, France. [2] Carney Institute for Brain Science, Dpt. of Cognitive Linguistic & Psychological Sciences, Brown University, Providence, RI 02912. [3] ELLIS Alicante, Spain. Correspondence to: Victor Boutin <victor_boutin@brown.edu>, Thomas Serre <thomas_serre@brown.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

[1] For each pair the human samples are (by row): right - left - left; right - right - left; right - right - left.

ticularly promising class of models for one-shot generation. Clever methods for conditioning on a context vector (Giannone & Winther, 2021) or using direct guidance from the exemplar (Ho & Salimans, 2022) has led to diffusion models that can produce near photo-realistic samples – consistently outperforming the state-of-the-art in one-shot image generation (see Section 2.2 for more details on one-shot diffusion models).

The ability of diffusion models to synthesize photo-realistic images speaks to their great expressivity – which begs the question: "Can diffusion algorithms also serve as effective models of the human creative process for sketch drawing task?". Addressing this query necessitates a departure from photo-realistic image generation, as humans are incapable of producing such images. Instead, retraining generative models on more human-compatible datasets becomes imperative. Therefore, we consider the one-shot drawing task as it offers a leveled playing field for both humans and machines.

In this article, we introduce QuickDraw-FewShot (QuickDraw-FS), a dataset built on the *Quick, Draw!* challenge (Jongejan et al., 2016), specifically designed for the few-shot image generation challenge. Building on the "diversity vs. recognizability" scoring framework by Boutin et al., 2022, we systematically compare humans with one-shot generation algorithms based on VAEs, GANs, and diffusion models. We show that diffusion models provide a better approximation of human drawing ability compared to VAEs and GANs. We further introduce the *originality* metric to measure the similarity between an individual sample and the corresponding exemplar. For a given sample, the originality score is computed as its ℓ_2 distance to the exemplar in the SimCLR latent space. We analyze the evolution of samples' recognizability as a function of their originality using *generalization curves*. Our results suggest that strengthening the guidance of diffusion models helps improve the humanness of their drawings, but they still fall short of approximating the originality and recognizability of human drawings. We further conduct an online psychophysics experiment to identify the most important image features for individual samples to be recognizable. Comparing these human-derived importance maps against those derived from diffusion models reveals a remarkable difference: humans tend to rely on fewer and more localized features than diffusion models. We suggest that these differences in visual strategies may be part of the reason for the remaining gap between machines and humans.

2. Related Works

2.1. Humans-Machines Comparison Frameworks

Researchers have historically used the Omniglot dataset to compare the generalization abilities of humans and machines on drawing tasks (Lake et al., 2019). To collect the Omniglot samples, human participants were presented with exemplars of novel handwritten characters and asked to reproduce them as accurately as possible (see Appendix 1 in (Lake et al., 2015)). A limitation of this experimental protocol is that it hinders human creativity by prompting subjects to literally copy the exemplar. In Appendix B, we confirm this limitation by comparing the distributions of intra-class variability for the Omniglot and our proposed dataset QuickDraw-FS (see Section 3.1 for more details on the datasets). Another limitation of the Omniglot dataset is that it contains a reduced number of samples per category ($n = 20$ samples). This prevents us from performing intra-class analyses that are statistically meaningful.

Different methods have been proposed to compare humans and machines in the one-shot generation task. Lake et al., 2015 use the visual Turing test in which participants are asked to distinguish human drawings from those generated by the models (similar to Figure 1). Another approach consists in asking human participants or classifiers (Lake et al., 2015) to evaluate the recognizability of individual samples. However, neither of these methods helps quantify the degree to which models are able to produce samples that are sufficiently diverse – at least compared to humans. More recently, a "diversity vs. recognizability" framework was proposed to circumvent this limitation via an additional critic network to evaluate samples' intra-class variability (Boutin et al., 2022). This diversity metric provides a single measure for an entire class, and hence, it does not provide the necessary granularity needed to evaluate individual samples. Here, we refine this diversity metric to systematically compare the originality of individual drawings produced by humans and models (Tiedemann et al., 2022).

2.2. One-Shot Generative Models

The one-shot image generation task involves synthesizing variations of a visual concept that has not been seen during training. Let $\mathbf{x} \in \mathbb{R}^D$ be the image data to model, and $\mathbf{y} \in \mathbb{R}^D$ be the exemplar. Mathematically, the task involves learning the conditional probability distribution $p(\mathbf{x}|\mathbf{y})$. Herein, we mainly focus on diffusion models to learn $p(\mathbf{x}|\mathbf{y})$ (Song & Ermon, 2019; Sohl-Dickstein et al., 2015), but VAEs (Kingma & Welling, 2013) or GANs (Goodfellow et al., 2014) models are also succinctly described afterwards.

A diffusion process describes the transformation of an observed data $\mathbf{x_0} \in \mathbb{R}^D$ to a pure noise $\mathbf{x}_T \in \mathbb{R}^D$ using a sequence of latent variables $\{\mathbf{x_i}\}_{i=1}^{T-1} \in \mathbb{R}^{D\times(T-1)}$. This transformation is parameterized by the approximated transition probability $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})$ (see Appendix E.1 for more mathematical details). The first diffusion model we consider in this article is the conditional Denoising Diffusion Probabilistic Model (DDPM), introduced by Ho et al., 2020.

The DDPM reduces the learning of $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t})$ to the optimization of a simple conditional auto-encoder ϵ_θ (with $\epsilon_\theta : \mathbb{R}^D \times \mathbb{R}^D \to R^D$, see Appendix E.2):

$$\underset{\theta}{\text{argmin}}\, \mathbb{E}_{(\mathbf{x_t},\mathbf{y}),\epsilon}\Big[\|\epsilon_\theta(\mathbf{x}_t,\mathbf{y})-\epsilon\|_2^2\Big] \text{ s.t. } \epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I}) \quad (1)$$

Said differently, ϵ_θ is trained to predict the noise ϵ using a degraded sample $\mathbf{x}_t$ and the exemplar $\mathbf{y}$. Equation (1) is a denoising score matching objective (Song et al., 2020), so the optimal model ϵ_{θ^*} matches the following score function:

$$\nabla_{\mathbf{x}_t}\log p_{\theta^*}(\mathbf{x}_t|\mathbf{y}) \approx -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\epsilon_{\theta^*}(\mathbf{x}_t,\mathbf{y}) \qquad (2)$$

In Equation (2), $\bar{\alpha}_t$ is used to schedule the noise degradation of $\mathbf{x}_t$ (see Equation (6) in Appendix E.1). Training information, details on the architecture, and samples generated by the DDPM are available in Appendices F and I.

Dhariwal & Nichol, 2021 have shown that one could improve the conditioning signal of the DDPM by guiding the forward process with a classifier. The second diffusion model we consider, the Classifier Free Guided Diffusion Model (CFGDM), adopts a similar idea but replaces the classifier with a conditional generative model (Ho & Salimans, 2022). The score function of the CFGDM can be expressed using the DDPM one (see Appendix G.1 for more details):

$$\begin{aligned}\nabla_{\mathbf{x}_t}\log p_{\theta^*,\gamma}(\mathbf{x}_t|\mathbf{y}) = (1+\gamma)\nabla_{\mathbf{x}_t}\log p_{\theta^*}(\mathbf{x}_t|\mathbf{y})\\ -\gamma\nabla_{\mathbf{x}_t}\log p_{\theta^*}(\mathbf{x}_t) \qquad (3)\end{aligned}$$

This formulation introduces a guidance scale γ to tune the part of the distribution that captures the influence of the conditioning signal. Note that in Equation (3), the two terms on the right hand side are parametrized by the same neural networks and are trained together (with $\nabla_{\mathbf{x}_t}\log p_{\theta^*}(\mathbf{x}_t) \propto \epsilon_{\theta^*}(\mathbf{x}_t,\varnothing)$ and $\nabla_{\mathbf{x}_t}\log p_{\theta^*}(\mathbf{x}_t|\mathbf{y}) \propto \epsilon_{\theta^*}(\mathbf{x}_t,\mathbf{y})$, see Appendix G.2). In the CFGDM, γ is set to 1, except if specified otherwise. Training information, details on the architecture, and samples generated by the CFGDM are available in Appendices G and J.

The third diffusion model we consider is the Few-shot Diffusion Model (FSDM, Giannone et al., 2022). In the FSDM, the feature maps of the auto-encoder ϵ_θ are conditioned with a context vector $\mathbf{c} = h(\mathbf{y})$ using a FiLM-like mechanism (Perez et al., 2017). Note that this is different from the DDPM and the CFGDM that are conditioned by stacking the degraded samples $\mathbf{x}_t$ with the exemplar $\mathbf{y}$. We refer the reader to Appendices H and N for more information on the conditioning mechanism, the architecture, and the samples of the FSDM.

For the sake of comparison, we also include in this study the one-shot generative models presented in Boutin et al., 2022: the VAE-NS, the VAE-STN, the DA-GAN-RN and the DA-GAN-UN. Both VAE-NS and VAE-STN belong to the family of conditional Variational Auto-Encoders (VAE). The VAE-NS, also called the Neural Statistician (Edwards & Storkey, 2016; Giannone & Winther, 2021) is conditioned on a *context* set (similar to FSDM, see Appendix K). The VAE-STN is a sequential VAE which includes an attention mechanism that learns to focus on important locations of the exemplar image (Rezende et al., 2016). The VAE-STN iteratively generates images using a recurrent network (see Appendix L). Both DA-GAN-RN and DA-GAN-UN are Data Augmentation Generative Adversarial Networks, that are conditioned on a compressed representation of the exemplar image (Antoniou et al., 2017). The DA-GAN-UN is based on the U-Net architecture and the DA-GAN-RN leverages the ResNet architecture (see Appendix M). The code to train these models and to reproduce all the results of this paper is available on `https://github.com/serre-lab/diffusion_as_artist`.

3. Methods

3.1. Datasets

Omniglot is composed of binary images representing $1,623$ classes of handwritten letters and symbols (extracted from 50 alphabets) with only 20 samples per class (Lake et al., 2015). We have downsampled the original dataset to be 50×50 pixels. In this article, we use the weak generalization split, in which the training set is composed of all available symbols minus 3 symbols per alphabet left aside for the test set (Rezende et al., 2016). It is called *weak* because all the alphabets are shown during the training (but not all symbols).

QuickDraw-FewShot (QuickDraw-FS) is built on the *Quick, Draw !* challenge (Jongejan et al., 2016), in which human subjects are presented with object names and are asked to draw them in less than 20 seconds. The original dataset is not suitable for the one-shot generation task because some object categories include more than one visual concept. For example, the 'clock' visual concept includes digital and analog clocks (see Figure A.1 for more examples). We use a clustering method on the original QuickDraw dataset to isolate distinct visual concepts for each object category (see Appendix A.2 for a step-by-step description of the clustering and the filtering method). The resulting dataset, called QuickDraw-FS, is fully compatible with the one-shot scenario. It is composed of black & white images representing 665 distinct visual concepts. Each of the visual concepts is described by an exemplar and 500 variations. The training set is made of 550 randomly sampled visual concepts. The remaining 115 visual concepts constitute the test set. We have downsampled the images to be 48×48 pixels so that it could be fed into ResNet blocks without resizing.

Note that the dissimilarity between the training and test visual concepts is higher in the QuickDraw-FS dataset than in the weak generalization split of the Omniglot dataset. Consequently, the one-shot generation task requires a greater generalization ability in the QuickDraw-FS dataset.

3.2. The Diversity vs. Recognizability Framework

The "diversity vs. recognizability" framework was initially proposed by Boutin et al., 2022 to evaluate the performance of humans and machines on the one-shot generation task. Within this framework, 2 distinct scores are assigned to the variations generated from a given exemplar: the diversity, which assesses the intra-class variability of all variations of a given class, and the recognizability, which measures whether the variations fall within the same class as the exemplar. Intuitively, an ideal model should fall in the top-right region of the diversity vs recognizability space (see Figure 1 of Boutin et al., 2022) : it should produce recognizable samples (high y-axis values) that are as diverse as possible (high x-axis values). On the contrary, a model that can only produce "copy" of the exemplar would fall in the top-left corner, and a model producing random nonsensical samples would be located on the bottom-right corner. The diversity score is computed with a standard deviation across samples from the same class (see Appendix P for more details). Note that the intra-class variability is not directly computed in the pixel space, but rather in the feature space of a SimCLR network (Chen et al., 2020). The SimCLR features offer greater invariance to transformations such as rotation, translation and scaling. We call "recognizability", the classification accuracy as evaluated by a one-shot classifier. Herein, we use a Prototypical Net to perform one-shot classification (Snell et al., 2017). The choice of the 2 critic networks (i.e. the SimCLR and the Prototypical) is validated and discussed in the Annex S3 of Boutin et al., 2022. On the QuickDraw-FS dataset, we have adapted the architectures of the critic networks (see Appendix C). In this work, we normalize the diversity metric such that the average standard deviation across features is equal to one. Such a normalization allows us to more faithfully compare the diversity and recognizability scores across different datasets or different critic networks (see Appendix D).

4. Results

4.1. Diversity vs Recognizability

Figures 2a and 2b show diversity vs. recognizability plots for all algorithms described in Section 2.2. We conducted an extensive hyper-parameter exploration with a total of 1 320 and 1 212 models trained for Omniglot and QuickDraw-FS, respectively. Each data point represents a single model whereby the diversity and recognizability scored were averaged over all classes of the test set. The black star in each plot corresponds to the human ideal model, and the colored points are the one-shot generative models. Large data points represent models' base architectures, with a comparable number of parameters: $\approx$ 6-7M parameters for Omniglot (Figure 2a) and $\approx$ 12-13M parameters for QuickDraw-FS (Figure 2b). The **VAE-NS**, the VAE-STN, the DA-GAN-UN and the DA-GAN-RN trained on Omniglot are the same exact architectures as reported in Boutin et al., 2022 using original code from these authors. The different hyper-parameters we have varied to obtain the point cloud for each model are described in Appendices F to N.

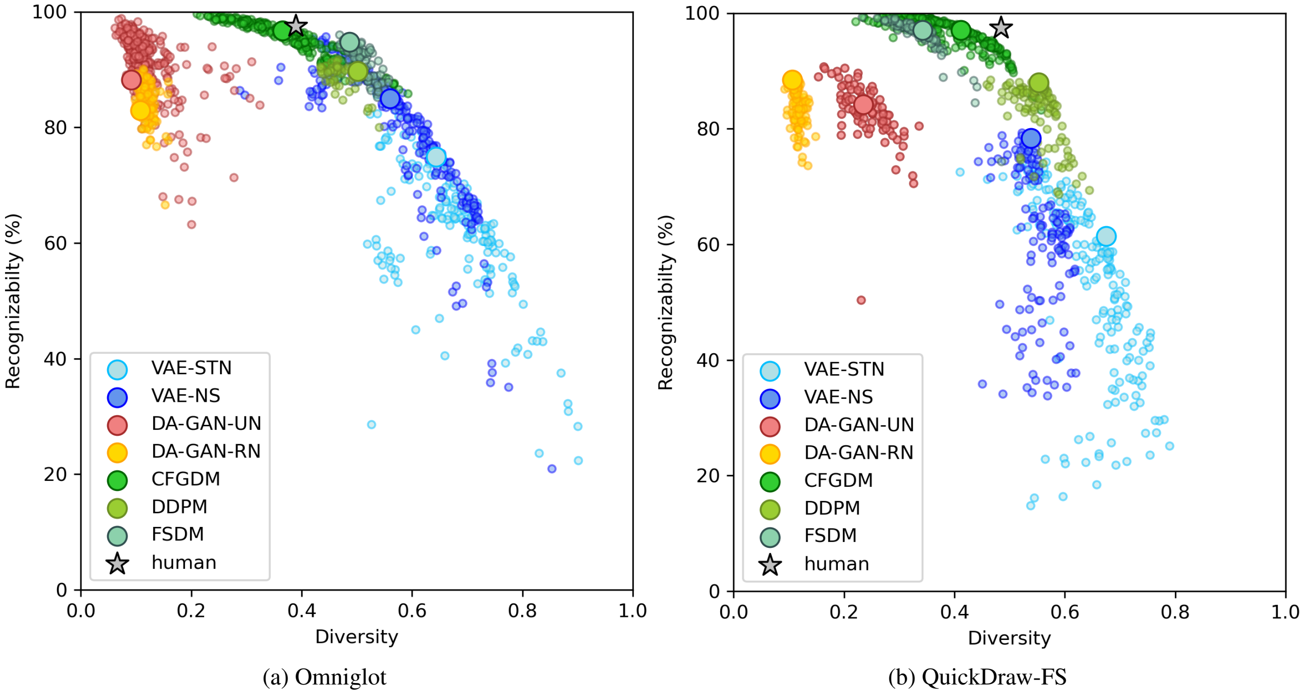
Overall, GANs (DA-GAN-RN and DA-GAN-UN) tend to exhibit low diversity for both the Omniglot (Figure 2a) and the QuickDraw-FS datasets (Figure 2b).VAEs (**VAE-NS** and VAE-STN) display a higher diversity but also slightly lower recognizability for both datasets. This observation has already been made on the Omniglot dataset by Boutin et al., 2022; Here, we generalize this result to a more complex dataset (QuickDraw-FS). Furthermore, we also observe a drop in recognizability between the VAEs trained on Omniglot and those trained on QuickDraw-FS: from 86% to 78% for the **VAE-NS** and from 74% to 61% for the VAE-STN. This phenomenon is less pronounced for GANs and diffusion models. This suggests that GANs (DA-GAN-RN, DA-GAN-UN) and diffusion models (**DDPM**, **CFGDM** and FSDM) are easier to scale up, without major architecture changes, to more complex datasets than VAEs (**VAE-NS** and VAE-STN). This observation tends to corroborate the scaling difficulties already reported for the VAEs (Bond-Taylor et al., 2021; Vahdat & Kautz, 2020).

We notice that with identical numbers of parameters to VAEs and GANs, diffusion models (**DDPM**, **CFGDM** and FSDM) can produce more recognizable samples on Omniglot and QuickDraw-FS. This observation aligns with the latest findings suggesting that diffusion models beat other models in terms of sample quality (Dhariwal & Nichol, 2021). In terms of diversity, the diffusion models are in between the GANs and the VAEs. The **CFGDM**, the **DDPM**, and FSDM data points consistently fall in a close neighborhood of the human ideal observer (black star) for both datasets. We conclude that diffusion models provide the best approximation of human-level drawings. Henceforth, we will focus on the diffusion models and the human data on the QuickDraw-FS dataset.

4.2. Generalization Curves

Here, we introduce the *originality* metric to quantify the distance between an individual sample and the corresponding exemplar. This distance is computed using a ℓ_2-norm in the

[2] `https://github.com/serre-lab/diversity_vs_recognizability`



(a) Omniglot

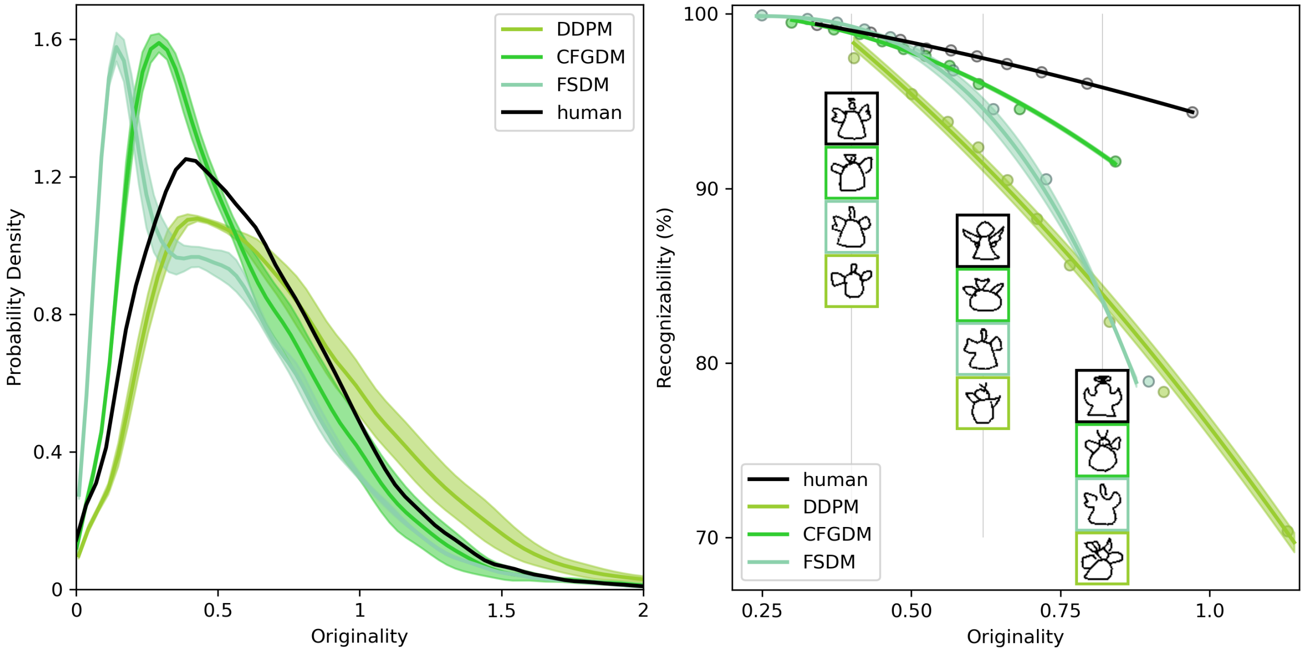
(b) QuickDraw-FS

Figure 2. Diversity vs. recognizability plots for models (colored data points) and humans (black/grey star) for **(a)** the Omniglot dataset (1 320 models tested) and **(b)** the QuickDraw dataset (1 212 models tested). Data points for **VAE-NS**, **VAE-STN**, **DA-GAN-RN** and **DA-GAN-UN** on Omniglot were computed using code from Boutin et al., 2022[2]. Each data point corresponds to the mean diversity and recognizability computed over all classes on the test set. Larger circles correspond to base architectures for which we controlled the number of parameters (≈ 6 − 7M for Omniglot and ≈ 12 − 13M for QuickDraw-FS). The **human** data point is computed based on the test samples of the Omniglot and the QuickDraw-FS datasets.

feature space of a SimCLR network (see Appendix C for more details on the SimCLR architecture). Intuitively, the higher the distance to the exemplar, the more original the sample and the higher the inventiveness and creativity of the corresponding model. We have validated our originality measure through a series of control experiments with different feature extractor networks and different distance metrics (see Appendix O). In Figure O.1, one can see that our originality metric is qualitatively similar to human judgments.

We draw the reader's attention to the fact that the originality and diversity metrics are different. Even though both metrics tell us something about the creative process of models, the ”diversity” is an aggregate per-class measure (the intra-class variability, hence one model as one value of diversity per class), while the ”originality” is a per-sample measure (the distance of one sample generated to the exemplar, hence we can probe the model at different levels of ”originality” for a given class). The former assesses the mean distance between samples of the category and the corresponding exemplar, and the latter evaluates the mean distance to the center of the category cluster (see Appendix P for more details). In Figure 3a, we plot the distribution of the samples' originality for the **human**, the **DDPM**, the **CFGDM** and the **FSDM** on the QuickDraw-FS dataset. The **FSDM** distribution is highly concentrated in the low-originality region, which suggests that the model tends to produce samples that are similar to the exemplar. On the contrary, the **DDPM** has a distribution spreading towards higher originality (positive skewness). This indicates that the **DDPM** has the ability to generate samples that are more dissimilar to the exemplar. The originality metric informs us about the inventiveness of the corresponding model, through the distance to the exemplar, but it does not tell us anything about how faithfully the sample represents the visual concept conveyed by the exemplar.

To overcome this limitation, we introduce the *generalization curve*. For a given model, the generalization curve quantifies the evolution of the recognizability at different levels of originality. It is important to emphasize that a generalization curve describes samples that are all generated by the same model. For each class, we sort the samples into originality bins, such that the samples belonging to the same bin have a relatively similar distance to the exemplar. In particular, we arrange the samples into 10 bins with 50 samples in each bin. As a result, the QuickDraw-FS dataset is split into 10 sub-sets, each containing samples with comparable originality levels. We then compute the average originality and recognizability for each bin (see data points



(a) Probability density

(b) Generalization curves

Figure 3. **(a)**: Distribution of the samples' *originality* computed for the **DDPM**, **CFGDM**, **FSDM** and **humans**. Originality scores were computed as the ℓ_2-distance, in the SimCLR feature space, between a sample and its exemplar. Distributions were estimated from histograms using a Gaussian density kernel estimation approach. **(b)**: Generalization curves for the **DDPM**, **CFGDM**, **FSDM** and **humans**. Each data point corresponds to the average originality and recognizability over the samples in each of the 10 originality bins. Plain lines are smooth interpolations (polynomial regression) between data points. Thumbnails show human and model-generated samples for 3 different levels of originality. For both panels, base architectures (corresponding to the larger markers in Figures 2a and 2b) trained on the QuickDraw-FS dataset were used for all models. Shaded areas are computed using the standard deviation over 3 different runs.

in Figure 3b). We ultimately derive generalization curves by smoothly interpolating between data points (using polynomial regression). For each model, we report the regression error in Appendix Q. These generalization curves are shown with plain lines for **human**, the **DDPM**, the **CFGDM** and the **FSDM** on the QuickDraw-FS dataset in Figure 3b. Note that such curves would not be statistically meaningful on the Omniglot dataset due to the reduced number of samples per class (20 samples). Intuitively, an agent with superior generalization capabilities would be expected to exhibit a generalization curve that is shifted towards the upper-right corner. Such an agent would be able to evaluate so accurately the decision boundary that it could produce samples lying in its close vicinity.

In Figure 3b, we observe that the **FSDM** model is able to produce samples with the highest recognizability ($\approx 100\%$) albeit with the lowest originality (≈ 0.25). The **FSDM** recognizability falls sharply as the originality score increases. The **DDPM** generalization curve spans the longest range in terms of originality (from 0.40 to 1.15) and recognizability (from 70% to 97%). The **DDPM** can produce samples that are different from the exemplar but are also poorly recognizable. The **CFGDM** samples are less original but also more recognizable than the **DDPM** samples. **Humans** maintain the best generalization curve in the high-originality regime: the **human** curve is above all others for originality values greater than 0.53. It suggests that **humans** can produce samples that are simultaneously more original and more recognizable than all the models. For an originality score of 0.75, the recognizability of **human** samples is 96%, that of **CFGDM** is around 92% and that of **DDPM** drops to 87%. Among all tested models, the **CFGDM** best approximates the human generalization curve.

In Figure 4, we study the impact of the guidance scale (the γ coefficient in Equation (3)) on generalization curves. Increasing the guidance scale has a double effect on the **CFGDM** score: i) it encourages the conditional term (first term of the RHS of Equation (3)) and ii) it penalizes the unconditional term (second term of the RHS of Equation (3)). When $\gamma = 0$, the unconditional term in Equation (3) disappears, the **CFGDM** score becomes then strictly equivalent to the **DDPM** one. The base architecture of the **CFGDM** is obtained with $\gamma = 1$. We observe improved recognizability and a decrease in originality as we increase the guidance scale from 0 to 2. This observation is even more pronounced for high originality values (see the right end of the curves in Figure 4 for different guidance scales). Overall, we observe a progressive shrinkage of the originality and the recogniz-

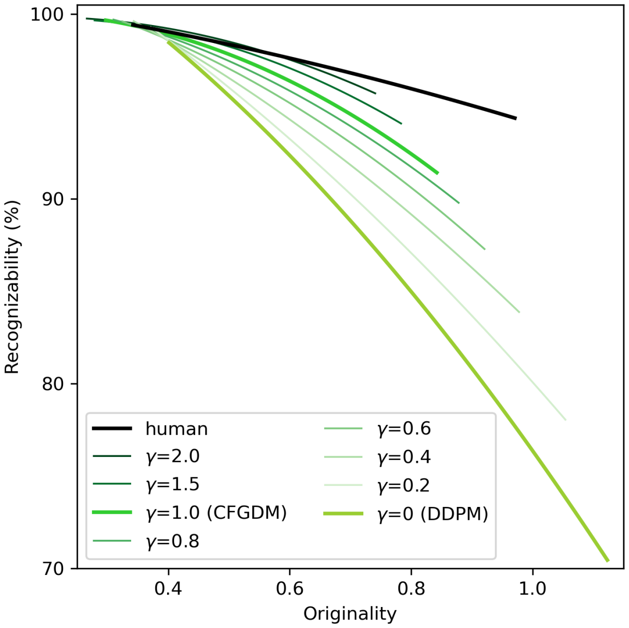


Figure 4. Generalization curves for **humans**, the DDPM and the CFGDM with different levels of guidance (γ) on the QuickDraw-FS dataset. Each curve represents a different model. The gray dashed line is the best possible generalization curve for the CFGDM models. For readability, we have omitted the data points (only smooth interpolation curves are shown; see Appendix Q for the corresponding interpolation errors).

ability range as we increase γ. Interestingly, a similar phenomenon has also been reported on natural images (Dhariwal & Nichol, 2021). The DDPM (i.e., when $\gamma = 0$) spans an originality range of 0.75 (from 0.4 to 1.15) and a recognizability range of 27% (from 70% to 97%) whereas the CFGDM, with $\gamma = 2$, spans an originality range of 0.5 (from 0.25 to 0.75) and a recognizability range of 3% (from 97% to 100%).

We note that the generalization curve of the CFGDM, with $\gamma = 2$, provides a good approximation to the **human** generalization curve for low originality values (below 0.6). But this model fails to account for **human** generalization in higher originality regimes. In Figure 4, we highlighted the best possible generalization curve for the CFGDM models with the gray dashed line. This curve is obtained by selecting the model with the highest recognizability for all originality levels. We observe that this curve still shows a severe drop in recognizability as the samples get more original. Among all tested models, we did not find one that is able to reach the recognizability of **humans** for a high level of sample originality.

4.3. Comparing Human and Machine Visual Features

To delve deeper into the differences observed between CFGDM and **humans**, we study the diagnosticity of individual features for each category.

We draw inspiration from attribution methods (Zeiler & Fergus, 2014; Sundararajan et al., 2017; Smilkov et al., 2017; Fel et al., 2021; Novello et al., 2022; Fel et al., 2022a) and use the score function decomposition of the CFGDM to visualize diagnostic features. We denote $\boldsymbol{\delta}_t^{\mathbf{y}}(\mathbf{x}, \mathbf{y})$ the part of the score conditioned on the exemplar at each time step t, and $\boldsymbol{\delta}_t(\mathbf{x})$ the unconditional part of the score. $\boldsymbol{\delta}_t^{\mathbf{y}}(\mathbf{x}, \mathbf{y})$ conveys information specific to the $\mathbf{y}$ exemplar, while $\boldsymbol{\delta}_t(\mathbf{x})$ encodes more general properties (e.g., background color, stroke size, etc). As previously observed in Figure 4, the misalignment between the conditional and unconditional signals is strongly related to image recognizability. Therefore, such a misalignment could be used to identify the most discriminative features. For a given sample $\mathbf{x}$ exemplified by the exemplar $\mathbf{y}$, we propose a metric, denoted $\phi(\mathbf{x}, \mathbf{y})$, to evaluate the misalignment at every position in the image.

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^{T} \left| \frac{\boldsymbol{\delta}_t^{\mathbf{y}}(\mathbf{x}, \mathbf{y})}{\|\boldsymbol{\delta}_t^{\mathbf{y}}(\mathbf{x}, \mathbf{y})\|_2} - \frac{\boldsymbol{\delta}_t(\mathbf{x})}{\|\boldsymbol{\delta}_t(\mathbf{x})\|_2} \right| \tag{4}$$

$$\text{s.t} \quad \begin{cases} \boldsymbol{\delta}_t^{\mathbf{y}}(\mathbf{x}, \mathbf{y}) & = \nabla_{\mathbf{x}_t} \log p_{\theta^*}(\mathbf{x}_t|\mathbf{y}) \\ \boldsymbol{\delta}_t(\mathbf{x}) & = \nabla_{\mathbf{x}_t} \log p_{\theta^*}(\mathbf{x}_t) \end{cases}$$

This metric is computed by accumulating over all time steps the absolute value of the difference between the normalized conditional and unconditional scores. For each category, we average over 10 misalignment maps to obtain the final feature importance map. In Figure 5a, we show representative feature importance maps for the CFGDM, trained on QuickDraw-FS, for 25 different categories (see Figure R.1 for more feature importance maps).

We conducted an online experiment, called ClickMe-QuickDraw, to get feature importance maps for comparison with **humans**. The ClickMe-QuickDraw experiment follows a similar protocol to the ClickMe experiment initially used by Linsley et al., 2018 to derive human importance maps for ImageNet. In ClickMe-QuickDraw, participants are asked to locate features in an image that they believe are important for categorizing it. As the participant selects important image regions, those regions are gradually revealed starting from a blank canvas and passed iteratively to a classifier. The participant gets rewarded whenever the classifier correctly classifies the canvas before the round time is up. At the end of each round, we obtain a ClickMe map: a map in which pixel intensities represent the probability of the pixel being painted by the participant. To obtain the importance feature map of a category, we average the ClickMe maps over all participants and images for that category. To keep a fair comparison with the CFGDM importance feature maps, the same images were used as those used to compute

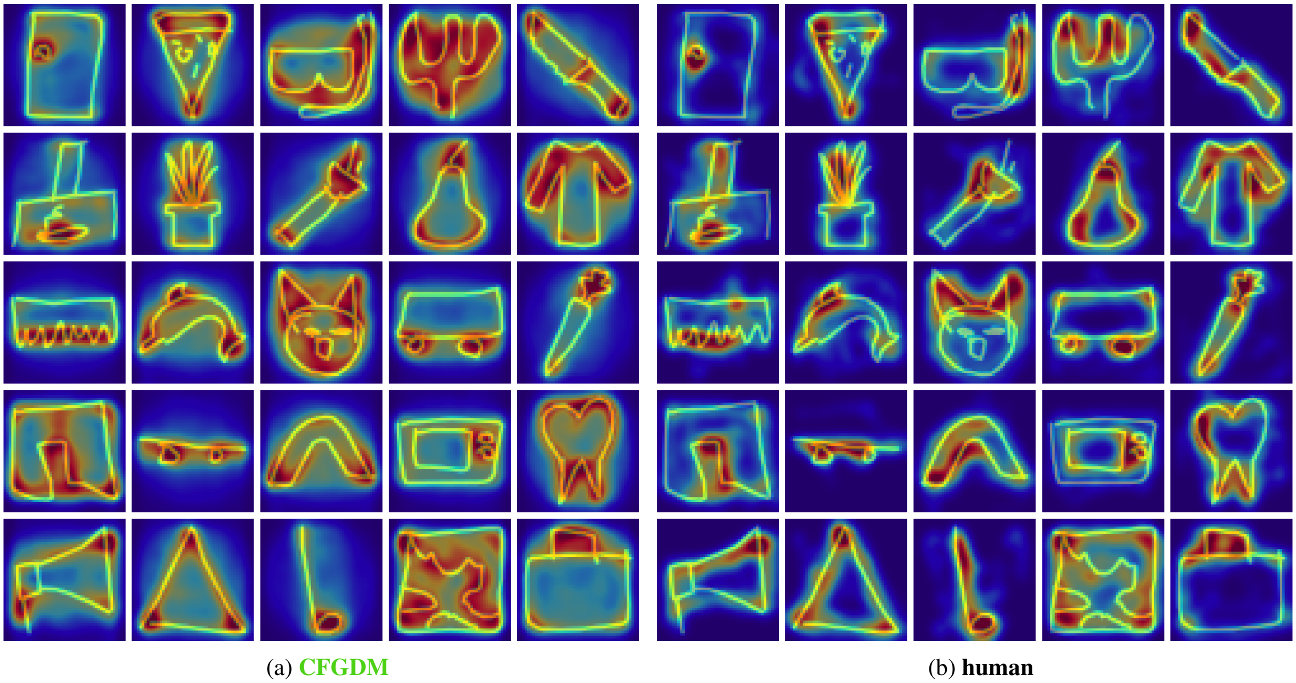


Figure 5. Importance maps (overlaid on exemplars) derived for **(a)** CFGDM and **(b)** **human** observers for 25 representative visual concepts. Hot vs. cold pixels indicate image locations that are more vs. less important. Maps for **(a)** CFGDM were obtained by averaging over $n = 10$ misalignment maps $\phi(\mathbf{x}, \mathbf{y})$ as defined in Equation (4). Maps for **human** observers were obtained using our ClickMe-QuickDraw online game.

the misalignment maps for the models. Crucially, previous studies have shown that the ClickMe experimental protocol produces feature importance maps that are perceptually meaningful (Linsley et al., 2017; 2018). For the ClickMe-QuickDraw experiment, we collected $1,050$ ClickMe maps from 102 participants. We refer the reader to Appendix S for more details on the ClickMe-QuickDraw experimental protocol as well as the statistics used to assess the reliability of the results. In Figure 5b, we show human feature importance maps for comparison with the model.

We observe that **CFGDM** importance maps are more diffuse than those of **humans**. The **CFGDM** gives importance (albeit weak) to the background in the close vicinity of the object while **humans** tend to focus only on sparser features of the object itself. In general, the category diagnostic features of **humans** are also highlighted in the **CFGDM** feature importance maps. In short, **humans** rely on fewer and more localized features to identify the object category. For example, **humans** consider that the ears are diagnostic of the "cat head" while the **CFGDM** tends to highlight the full head contour (3^{rd} row, 3^{rd} column in Figure 5). One interesting exception is "golf club", the **CFGDM** emphasizes the club's head while **humans** consider that the club's shaft is also important. The comparison between the **CFGDM** and **humans** feature importance maps suggests that they rely on different visual strategies.

5. Discussion

In this article, we compared humans and machines on a one-shot drawing task. We extended the "diversity vs. recognizability" framework of Boutin et al., 2022 to compare samples produced by various generative models against those produced by humans. We found that diffusion models (**DDPM**, **CFGDM** and **FSDM**) offer a better approximation to human drawings than VAEs (**VAE-NS**, **VAE-STN**) and GANs (**DA-GAN-RN**, **DA-GAN-UN**). Other studies have also reported state-of-the-art performances for diffusion models (Chahal, 2022; Dhariwal & Nichol, 2021; Peebles & Xie, 2022). We hypothesize that this success comes from the fact that diffusion models have to evaluate a simpler mathematical object than VAEs and GANs. The former learns the progressive transition between 2 noisy states, while VAEs and GANs have to encode the direct mapping from pure noise to the data distribution.

We further introduced the *originality* metric to evaluate the distance to exemplar for each individual sample. We found that the **CFGDM** provides a better approximation to human drawings in low-originality regimes compared to the **DDPM** and the **FSDM**. One unique feature of the **CFGDM** is that it penalizes the features that are common to all classes in favor of more diagnostic features. Interestingly, forcing the model to move further away from non-category specific features improves the humanness of generated drawings in a low-originality regime (see Figure 4). This suggests that humans might rely on a few features that are strongly discriminative to solve the one-shot drawing task. This hypothesis seems to align with the human data we have collected with the ClickMe-QuickDraw experiment: human feature importance maps tend to be sparser and tend to emphasize strongly localized features (see Figure 5b). Additionally, this result is in line with other psychophysics experiments that have shown that small but specific fragments of an image are sufficient to humans for correct categorization (Hegdé et al., 2008; Ullman et al., 2002).

Nevertheless, a question remains: how can humans produce drawings that are both highly original and recognizable? We speculate that this aspect of human drawings could be the consequence of their attentional strategy. Since humans seem to focus on a few localized features, non-important features may vary more freely to provide room for creativity. This hypothesis is supported by recent human experiments that have suggested that the alteration of non-discriminative features has little to no impact on their recognizability (Tiedemann et al., 2022).

Recent research has established the exceptional zero-shot generalization capabilities of diffusion models guided by expressive language embeddings (Saharia et al., 2022; Ramesh et al., 2021; Nichol et al., 2021; Rombach et al., 2022). Nevertheless, psychophysics studies have shown that humans demonstrate impressive generalization abilities in drawing tasks even when the drawings are semantically nonsensical (Tiedemann et al., 2022). It suggests that semantic knowledge is not a prerequisite for human to achieve good generalization, and our results suggest a similar pattern in one-shot generative models. However, given the inherently compositional nature of language, we speculate that semantic knowledge may greatly enhance the model's generalization capabilities.

The metrics we have leveraged in this work provide a rigorous framework to probe the humans and machines generalization abilities in the one-shot drawing task. Those metrics have been carefully designed to avoid the pitfalls that are inherently related to the task: the law-data regime, and the dissimilarity between the training and testing visual concepts (see section 2.1 in Boutin et al., 2022). Nevertheless, the evaluation metrics are based on critic networks (e.g. Prototypical Net or SimCLR) that might not be well aligned with the human perception. In future work, we plan to circumvent this issue using human-alignment methods (Fel et al., 2022b; Muttenthaler et al., 2022).

By introducing and quantifying samples' recognizability and originality, we wanted to shed light on the relationship between generalization and creativity. We hope the generalization curves of the types introduced in this work may help provide better benchmarks for future models and help further close the gap between machines and humans.

Aknowledgement

We thank Roland W. Fleming and his team for the insightful feedback and discussion about the diversity vs. recognizability framework. This work was funded by ANITI (Artificial and Natural Intelligence Toulouse Institute) and the French National Research Agency, under the grant agreement number : ANR-19-PI3A-0004. Additional funding was provided by ONR (N00014-19-1-2029) and NSF (IIS-1912280 and EAR-1925481). Computing hardware supported by NIH Office of the Director grant S10OD025181 via the Center for Computation and Visualization (CCV). J.C. has been partially supported by funding from a grant by Banco Sabadell Foundation and from the Valencian Government (Conselleria d'Innovació, Universitats, Ciència i Societat Digital) by virtue of a 2022 grant agreement (convenio singular 2022) and (Conselleria d'Innovació, Industria, Comercio y Turismo, Direccion General para el Avance de la Sociedad Digital) by virtue of a 2023 grant agreement (convenio singular 2023).

References

Antoniou, A., Storkey, A., and Edwards, H. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *arXiv preprint arXiv:2103.04922*, 2021.

Boutin, V., Singhal, L., Thomas, X., and Serre, T. Diversity vs. recognizability: Human-like generalization in one-shot generative models. *arXiv preprint arXiv:2205.10370*, 2022.

Chahal, P. Exploring transformer backbones for image diffusion models. *arXiv preprint arXiv:2212.14678*, 2022.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

Donald, M. *Origins of the modern mind: Three stages in the evolution of culture and cognition*. Harvard University Press, 1991.

Edwards, H. and Storkey, A. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Fel, T., Cadene, R., Chalvidal, M., Cord, M., Vigouroux, D., and Serre, T. Look at the variance! efficient black-box explanations with sobol-based sensitivity analysis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 26005–26014, 2021.

Fel, T., Ducoffe, M., Vigouroux, D., Cadène, R., Capelle, M., Nicodème, C., and Serre, T. Don't lie to me! robust and efficient explainability with verified perturbation analysis. *arXiv preprint arXiv:2202.07728*, 2022a.

Fel, T., Felipe, I., Linsley, D., and Serre, T. Harmonizing the object recognition strategies of deep neural networks with humans. *arXiv preprint arXiv:2211.04533*, 2022b.

Feldman, J. The structure of perceptual categories. *Journal of mathematical psychology*, 41(2):145–170, 1997.

Giannone, G. and Winther, O. Hierarchical few-shot generative models. *arXiv preprint arXiv:2110.12279*, 2021.

Giannone, G., Nielsen, D., and Winther, O. Few-shot diffusion models. *arXiv preprint arXiv:2205.15463*, 2022.

Goodenough, F. L. *Measurement of intelligence by drawings*. World Book Company, 1926.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hegdé, J., Bart, E., and Kersten, D. Fragment-based learning of visual object categories. *Current Biology*, 18(8):597–601, 2008.

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. *International Conference of Learning Representation, 2017*, 2016.

Ho, J. and Salimans, T. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Jongejan, J., Rowley, H., Kawashima, T., Kim, J., and Fox-Gieg, N. The quick, draw!-ai experiment. *Mount View, CA, accessed Feb*, 17(2018):4, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Koppitz, E. M. *Psychological evaluation of children's human figure drawings*. Grune & Stratton, 1968.

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

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, 2019.

Lee, S. H., Lee, S., and Song, B. C. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.1349*, 2021.

Li, R., Su, J., Duan, C., and Zheng, S. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv preprint arXiv:2007.14902*, 2020.

Linsley, D., Eberhardt, S., Sharma, T., Gupta, P., and Serre, T. What are the visual features underlying human versus machine vision? In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2706–2714, 2017.

Linsley, D., Shiebler, D., Eberhardt, S., and Serre, T. Learning what and where to attend. *arXiv preprint arXiv:1805.08819*, 2018.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Muttenthaler, L., Dippel, J., Linhardt, L., Vandermeulen, R. A., and Kornblith, S. Human alignment of neural network representations. *arXiv preprint arXiv:2211.01201*, 2022.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Novello, P., Fel, T., and Vigouroux, D. Making sense of dependence: Efficient black-box explanations using dependence measure. *arXiv preprint arXiv:2206.06219*, 2022.

Peebles, W. and Xie, S. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. Film: Visual reasoning with a general conditioning layer. *AAAI 2018 arXiv:1709.07871*, 2017.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.

Rezende, D., Danihelka, I., Gregor, K., Wierstra, D., et al. One-shot generalization in deep generative models. In *International conference on machine learning*, pp. 1521–1529. PMLR, 2016.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Ryan-Wenger, N. Use of children's drawings for measurement of developmental level and emotional status. *Journal of child and family nursing*, 4(2):139–149, 2001.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Tenenbaum, J. B. *A Bayesian framework for concept learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

Tiedemann, H., Morgenstern, Y., Schmidt, F., and Fleming, R. W. One-shot generalization in humans revealed through a drawing task. *Elife*, 11:e75485, 2022.

Ullman, S., Vidal-Naquet, M., and Sali, E. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, 2002.

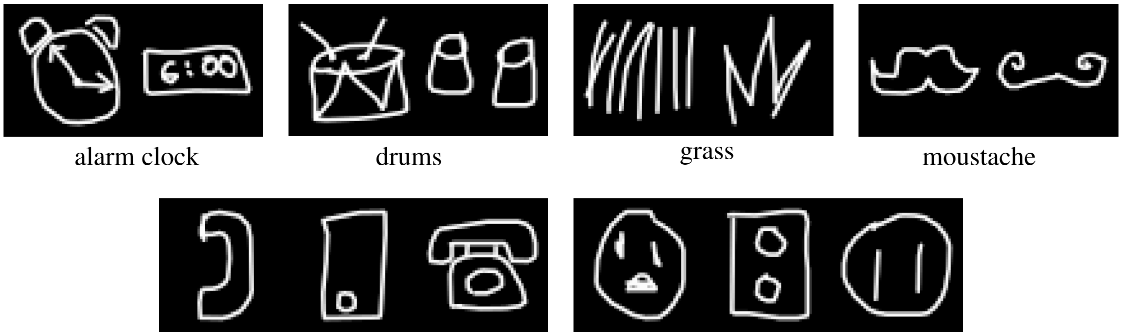
Ullman, T. D. and Tenenbaum, J. B. Bayesian models of conceptual development: Learning as building models of the world. *Annual Review of Developmental Psychology*, 2020.

Vahdat, A. and Kautz, J. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

A. Construction of the QuickDraw-FS dataset

A.1. Different visual concepts for the same object category


alarm clock

drums

grass

moustache

telephone

power outlet

Figure A.1. Examples of distinct visual concepts belonging to the same object category

In the *Quick, Draw !* challenge, participants have to draw objects belonging to a specific object category in less than 20 seconds (Jongejan et al., 2016). The dataset represents 345 object categories with approximately $150,000$ samples per category. The total number of drawings in the dataset exceeds 50 million. The participant is instructed on the object category using words, the object category actually includes more than one unique visual concept. We illustrate this phenomenon in Figure A.1 with some object categories composed of more than one visual concept. This property of the original QuickDraw dataset makes it incompatible with the one-shot image generation task because it requires all the samples of a given category to represent the same visual concept.

A.2. QuickDraw-FS processing steps

To circumvent this issue, we have created a new dataset, called QuickDraw-FewShot (QuickDraw-FS), built on the drawings of the *Quick, Draw !* challenge. More specifically, we have re-defined the original QuickDraw object categories so that they correspond to unique visual concepts. Here are the steps we have followed to create the QuickDraw-FS dataset :

1. We split the original QuickDraw dataset in a training set made of $5\,175\,000$ drawings (i.e., 345 categories, $15\,000$ samples each) and a testing set composed of $1\,725\,000$ samples (i.e., 345 categories, $5\,000$ samples each).

2. We train a SimCLR feature extractor on the QuickDraw training set. We refer the reader to Table 1 in Appendix C for more details on the SimCLR architecture.

3. For each object category, we project all the testing samples in the feature space of the SimCLR network.

4. We then apply a K-Means clustering algorithm on the features extracted previously. More specifically, we extract 6 clusters per object category

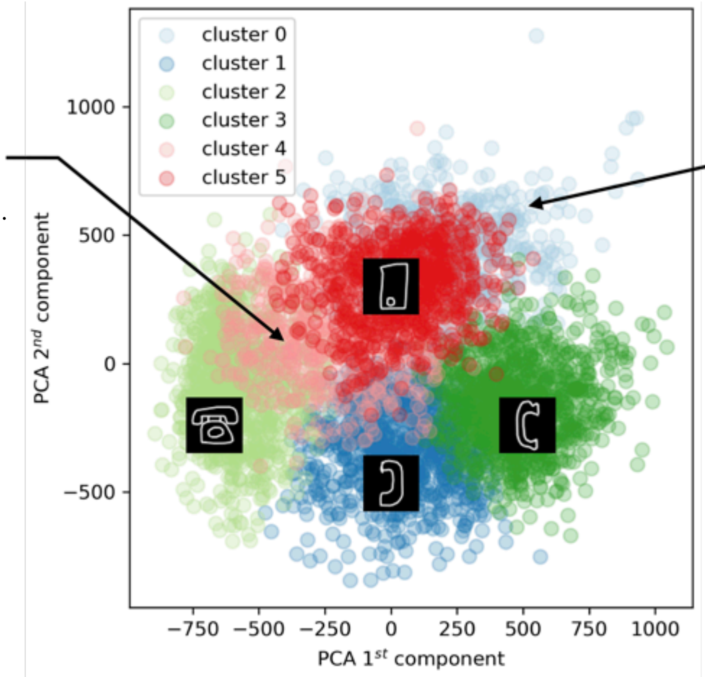
5. We filter out the clusters with less than 500 samples. This filter prevents us to choose clusters that are not representative enough of the object category.

6. We filter out the clusters with the largest spreading. The spreading size is computed as the mean ℓ_2-distance between the samples and the cluster center. When the spreading size is above $1\,800$, the cluster is filtered-out. This filter allows us to discard the *junk* clusters, composed of exuberant drawings that are all very different from each other. Note that this filter is triggered very occasionally.

7. We discard the clusters with centers that are not distant enough from the centers of other clusters. We do so by imposing a minimum ℓ_2-distance between clusters (set to 700). This rule prevents us to select 2 clusters that represent the same visual concept.

8. For each cluster, we pick an exemplar. The exemplar is selected as being the closest sample to the center of the cluster.

The filtering and clustering methods described in bullet points 3 to 8 are repeated for all object categories of the original QuickDraw dataset. Figure A.2 illustrates the selection process for a single object category.



The cluster 4 has been filtered-out because it is too close to other cluster' centers. See bullet point 7.

The cluster 0 has been filtered-out because it is too wide.
See preprocessing step 6.

Figure A.2. Illustration of the cluster selection process for the samples of the *phone* object category. The plot represents the PCA coordinates of the samples in the SimCLR feature space. Two clusters were filtered out by the selection process (clusters 4 and 0). The center of the remaining clusters corresponds to the exemplar of distinct visual concepts, as illustrated by the thumbnails.

At the end of the cluster selection process, we perform a visual inspection in which we have filtered out 2 *junk* clusters (see bullet point 6). The obtained QuickDraw-FS dataset is composed of 332 000 drawings (500 samples for each of the 665 distinct visual concepts). The train set is obtained by randomly sampling 550 visual concepts. The remaining visual concepts constitute the test set. In Figure A.3, we showcase randomly selected samples and their corresponding exemplar.

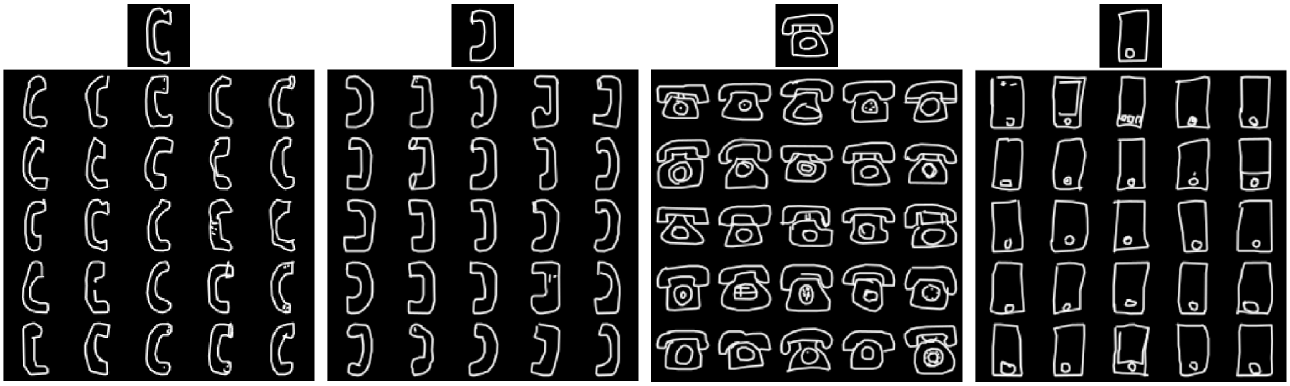


Figure A.3. Samples and exemplars of the distinct visual concepts extracted through the cluster selection process of the QuickDraw-FS dataset. The top thumbnails represent the exemplars describing the different visual concepts. The exemplars are picked so that they are approximately located at the center of the clusters (in the SimCLR feature space). The 5×5 grid of images showcases variations of the corresponding visual concepts. Those variations have been randomly sampled in the clusters.

B. Comparison between Omniglot and QuickDraw-FS

Herein we compare the intra-class variability of the Omniglot samples with the intra-class variability of the QuickDraw-FS samples. We refer the reader to Appendix A for more information on the QuickDraw-FS dataset. We show the distributions of the intra-class variability in Figure B.1. To obtain such a distribution, we (i) pass the samples into a feature extractor network (a SimCLR network), and (ii) compute the standard deviation for samples belonging to the same class. To have a faithful comparison between the 2 datasets, we normalized the features vector so that the standard deviation, across features, is set to one (see Appendix D for the reasons of such a normalization).

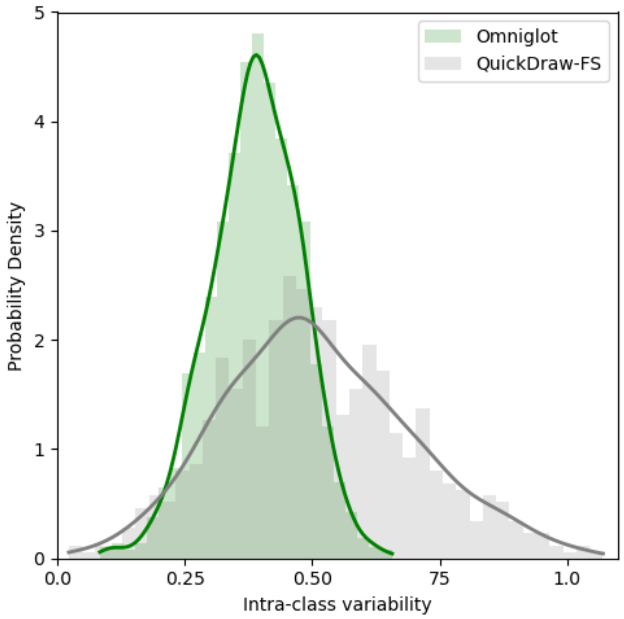


Figure B.1. Probability density of the (normalized) intra-class variability of the Omniglot (green) and the QuickDraw-FS dataset (grey). The intra-class variability is computed in the latent space of a SimCLR network, as the intra-class standard deviation.

We observe that the grey distribution covers a wider range of intra-class variability compared to the green distribution. The average intra-class variability is 0.37 for Omniglot and 0.48 for QuickDraw-FS. The maximum intra-class variability is 0.66 for Omniglot and 1.10 for QuickDraw-FS. It suggests that (i) the QuickDraw-FS samples are more diverse than those of the Omniglot dataset and (ii) the Omniglot samples do not reflect the human ability to produce original and diverse drawings.

This phenomenon could be explained by the way the Omniglot samples have been collected. Participants are presented with category exemplars and asked to draw, as accurately as possible, a replica of the exemplar (see Appendix 1 in Lake et al., 2015). This experimental protocol implicitly reduces the variability of the samples. For the QuickDraw dataset, the participants are presented with a word describing the object and are asked to draw the objects without other constraints. Even if the filtering process we have performed to obtain the QuickDraw-FS dataset should reduce the samples' intra-class variability (see Appendix A), it is still higher than the Omniglot intra-class variability.

C. Diversity vs Recognizability framework for the QuickDraw-FS dataset

The diversity versus recognizability framework leverages 2 critic networks to i) extract the features to compute the diversity metric, and ii) evaluate the one-shot classification accuracy for the recognizability. Similarly to Boutin et al., 2022, we use a SimCLR network (Chen et al., 2020) to extract features and we leverage a Prototypical Net (Snell et al., 2017) for the computation of the recognizability. We have increased the size of the critics' network architecture compared to Boutin et al., 2022 to adapt to the complexity of the QuickDraw-FS dataset. Here we describe the new architectures of both the SimCLR and Prototypical Network

C.1. SimCLR on QuickDraw-FS

Table 1 describes the architecture of the SimCLR network (Chen et al., 2020) trained on the QuickDraw-FS dataset. We use the Pytorch convention to describe the layers of the network.

Table 1. Description of the SimCLR Architecture

Network	Layer	# params
ConvBlock(In$_c$, Out$_c$)	Conv2d(In$_c$, Out$_c$, 3, padding=1)	In$_c$ $\times$ Out$_c$ $\times$ 3 $\times$ 3 + Out$_c$
	BatchNorm2d(Out$_c$)	2 x Out$_c$
	ReLU	-
	MaxPool2d(2, 2)	-
SimCLR	ConvBlock(1, 64)	0.7 K
	ConvBlock(64, 128)	74 K
	ConvBlock(128, 128)	149 K
	Flatten	-
	ReLU	-
	Linear(4608, 256)	1 179 K
	ReLU	
	Linear(256, 128)	32 K

The overall number of parameters of the SimCLR network we are using is around 1.4 M parameters. The features are extracted on the first fully-connected layer after the last convolutional layer (i.e., of size 256).

All other implementation details (augmentation, training parameters) are the same as those described in Appendix S2 of Boutin et al., 2022.

C.2. Prototypical Net on QuickDraw-FS

For the Prototypical Net trained on QuickDraw-FS, we leverage a ResNet-like architecture (He et al., 2016). This architecture is described in Table 2.

Table 2. Description of the Prototypical Net Architecture

Network	Layer	# params
ResNetBlock(In$_c$, Out$_c$)	Conv2d(In$_c$, Out$_c$, 3, padding=1)	In$_c$ $\times$ Out$_c$ $\times$ 3 $\times$ 3 + Out$_c$
	BatchNorm2d(Out$_c$)	2 x Out$_c$
	ReLU	-
	Conv2d(Out$_c$, Out$_c$, 3, padding=1)	Out$_c$ $\times$ Out$_c$ $\times$ 3 $\times$ 3 + Out$_c$
	BatchNorm2d(Out$_c$)	2 x Out$_c$
	## Shortcut connection	
	Conv2d(In$_c$, Out$_c$, 1, padding=1)	In$_c$ $\times$ Out$_c$ + Out$_c$
	BatchNorm2d(Out$_c$)	2 x Out$_c$
	ReLU	-
Prototypical Net	Conv2d(1, 64, 3, padding=1)	0.6 K
	BatchNorm2d(128)	0.12 K
	ReLU	-
	ResNetBlock(64, 64)	78 K
	ResNetBlock(64, 64)	78 K
	ResNetBlock(64, 128)	230 K
	ResNetBlock(128, 128)	312 K
	ResNetBlock(128, 256)	919 K
	ResNetBlock(256, 256)	1 246 K
	ResNetBlock(256, 512)	3 673 K
	ResNetBlock(512, 512)	4 984 K
	AvgPool2d(6, 6)	-
	Linear(512, 256)	131 K
	Linear(256, 128)	32 K

The overall number of parameters of the Prototypical Net we are using is around 11.6 M parameters. The loss of the Prototypical Net is applied to the output of the last fully connected layers (of size 128).

To prevent over-fitting and to adapt to the variability of the QuikDraw-FS dataset, we have randomly applied the following augmentation: first a horizontal flip, then a vertical flip, and last an affine transformation. The affine transformation is a combination of a rotation (with an angle randomly selected in the range $[-180°, 180°]$), a translation (randomly selected in $[-10\text{px}, 10\text{px}]$), a zoom (with a ratio randomly selected $[0.5, 1.5]$). Note that the augmentation is applied similarly to all the samples belonging to the same class so that it is virtually increasing the number of classes of the dataset.

All other training parameters are similar to those described by Boutin et al., 2022 in Appendix S1.

The code for training both critic networks on the QuickDraw-FS dataset is available online at `https://anonymous.4open.science/r/Diffusion_vs_human/`.

D. Effect of the normalization on the diversity metric

The diversity is obtained by computing a dispersion metric in the feature space. More specifically we use a Bessel-corrected standard deviation, applied in the latent space of a SimCLR network (Chen et al., 2020). This is the exact same setting as the one described in Boutin et al., 2022.

This way of computing the diversity metric has 2 main drawbacks: i) it is unbounded, and ii) it depends on the image size and on the size of the feature space of the SimCLR network. If we compare models on the same dataset, with a unique setting of the SimCLR network, those limitations are not problematic. But those drawbacks prevent us to compare diversity values on different datasets (and thus different SimCLR settings).

To circumvent these problems, we normalize the values of the feature vector so that its standard deviation (in the feature space) is set to one for each individual sample. This is important to note that this normalization is performed using a standard deviation computed along the features coordinate. In this case, the standard deviation quantifies the dispersion of the feature activation. In the calculation of the diversity value, we also use the standard deviation, but that one is computed along the sample axis. Consequently, the standard deviation used to compute the diversity quantifies the dispersion of the sample (in a given category).

We run a control experiment to verify that the proposed feature normalization is not changing the model's relative position on the diversity axis. To do so, we plot the models' diversity when the features are normalized (x-axis) or not (y-axis) (see Figure D.1). We report a linear correlation of $R^2 = 0.99$ and a Spearman rank-order correlation of $\rho = 0.99$.

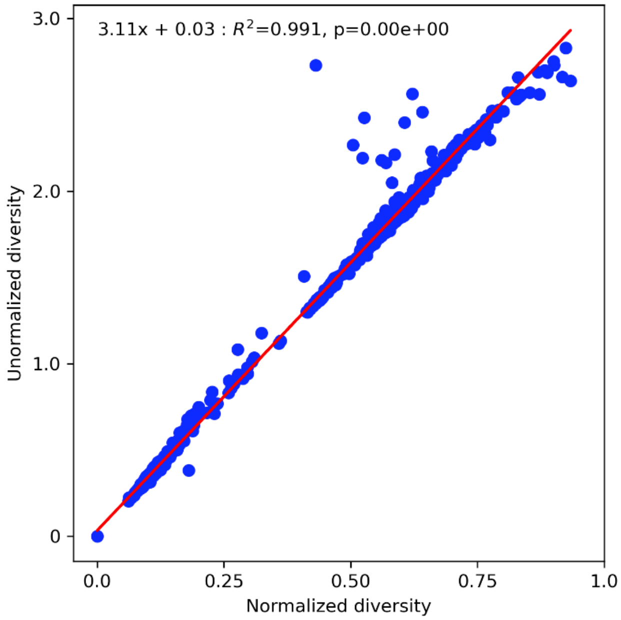


Figure D.1. Control experiment to compare the effect of feature normalization in the computation of the diversity value. Each data point corresponds to the mean diversity for a single model. In this graph, we have included models trained on Omniglot and QuickDraw-FS.

The high linear correlation and Spearman rank-order correlation suggest that the normalization operation is not changing the models' relative position on the diversity axis. Therefore, the proposed normalization method allows us to i) maintain the relative position of the model within a given SimCLR setting, and ii) compare models evaluated with different SimCLR settings.

E. Mathematics behind diffusion process

E.1. Diffusion process parametrization

Herein we detail the mathematics behind the diffusion models. Most of the demonstrations below are inspired by other works (Song & Ermon, 2019; Sohl-Dickstein et al., 2015; Ho et al., 2020) and are adapted to the few-shot image generation scenario. Even though those mathematical derivations are not crucial for a good understanding of our work, we include them to make sure our article is self-contained and complete.

A diffusion process describes the transformation of a pure noise $\mathbf{x}_T \in \mathbb{R}^D$ to an observed data $\mathbf{x_0} \in \mathbb{R}^D$ through a sequence of latent variables $\{\mathbf{x_i}\}_{i=1}^{T-1} \in \mathbb{R}^{D\times(T-1)}$. Diffusion models include a forward process modeling the transition probability $p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{y})$ and a reverse process that parameterize $q(\mathbf{x_t}|\mathbf{x_{t-1}})$. The directed graphical model under consideration is shown in Figure E.1.

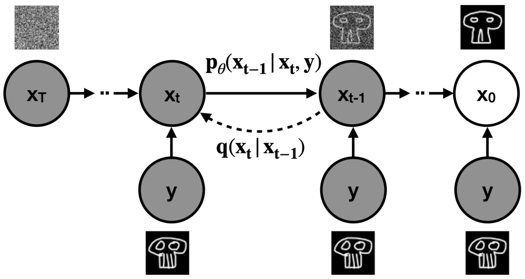

$\mathbf{p}_\theta(\mathbf{x_{t-1}}\,|\,\mathbf{x_t}, \mathbf{y})$
$\mathbf{x_T}$
$\mathbf{x_t}$
$\mathbf{x_{t-1}}$
$\mathbf{x_0}$
$\mathbf{q}(\mathbf{x_t}\,|\,\mathbf{x_{t-1}})$
y
y
y

Figure E.1. The directed graphical model considered in this work. Dotted and plain arrows represent the forward and reverse processes, respectively. The random variable $\mathbf{y}$ is exemplified by the skull exemplar (see bottom thumbnails), and the $\mathbf{x_i}$ latent variables are exemplified using skull samples with varying noise levels (see top thumbnails).

The forward process is parametrized as follow (Ho et al., 2020):

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \text{with} \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \text{s.t.} \quad \{\beta_t \in (0,1)\}_{i=1}^{T} \tag{5}$$

In Equation (5), β_t controls the step size of the diffusion process. Using the successive product of Gaussian, one can reparametrize $\mathbf{x}_t$ to express it without referring to the intermediate latent variables $\{\mathbf{x_i}\}_{i=1}^{t-1}$:

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2}\sqrt{1-\alpha_t\alpha_{t-1}}\epsilon \\ &= ... \\ &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \quad \text{with} \quad \alpha_t = 1-\beta_t \quad \text{and} \quad \bar{\alpha}_t = \prod_{i=1}^{t}\alpha_t \end{aligned} \tag{6}$$

Consequently, we have:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}) \tag{7}$$

The reverse process, also called the generative process, is conditioned on the exemplar $\mathbf{y}$ to recover the data from the noise (Ho et al., 2020):

$$p_\theta(\mathbf{x}_{0:T}|\mathbf{y}) = p_\theta(\mathbf{x}_T|\mathbf{y}) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) \quad \text{with} \quad \left\{ \begin{array}{ll} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y}) & = \mathcal{N}(\mathbf{x}_t; \mu_\theta(\mathbf{x}_t, t, \mathbf{y}), \sigma_t^2\mathbf{I}) \\ p_\theta(\mathbf{x}_T|\mathbf{y}) & = p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{array} \right. \tag{8}$$

E.2. From variational lower bound to auto-encoder optimization

We first express the Variational Lower Bound of the diffusion model using Jensen's inequality (Ho et al., 2020):

$$
\begin{aligned}
\mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0|\mathbf{y}) &= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)} \log \big(\int p_\theta(\mathbf{x}_{0:T}|\mathbf{y}) d\mathbf{x}_{1:T} \big) \\
&= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)} \log \big(\int q(\mathbf{x}_{1:T}|\mathbf{x}_0) \frac{p_\theta(\mathbf{x}_{0:T}|\mathbf{y})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} d\mathbf{x}_{1:T} \big) \\
&= \mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)} \log \Bigg(\mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big[\frac{p_\theta(\mathbf{x}_{0:T}|\mathbf{y})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big] \Bigg) \\
&\leq \mathbb{E}_{\mathbf{x}_{0:T}\sim q(\mathbf{x}_{0:T})} \log \Big(\frac{p_\theta(\mathbf{x}_{0:T}|\mathbf{y})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \Big) = -L_{VLB}
\end{aligned}
$$

The Variational Lower Bound could be written as a sum of KL terms (Sohl-Dickstein et al., 2015):

$$
\begin{aligned}
L_{VLB} &= \mathbb{E}_q \Big[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T}|\mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[\log \frac{\prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p(\mathbf{x}_T|\mathbf{y}) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} \Big] \quad \text{using Eq. (5) and (8)} \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T|\mathbf{y}) + \sum_{t=1}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T|\mathbf{y}) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T|\mathbf{y}) + \sum_{t=2}^{T} \log \Big(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} \Big) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T|\mathbf{y}) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} + \sum_{t=2}^{T} \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[-\log p_\theta(\mathbf{x}_T|\mathbf{y}) + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} + \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y})} \Big] \\
&= \mathbb{E}_q \Big[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T|\mathbf{y})} + \sum_{t=2}^{T} \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y}) \Big] \\
&= \mathbb{E}_q \Bigg[KL\big[q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T|\mathbf{y})\big] + \sum_{t=2}^{T} KL\big[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})\big] - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y}) \Bigg] \\
&= \sum_{t=0}^{T} L_t \quad \text{with} \quad \begin{cases} L_0 &= -\mathbb{E}_q \Big[\log p_\theta(\mathbf{x}_0|\mathbf{x}_1, \mathbf{y}) \Big] \\ L_t &= \mathbb{E}_q \Big[KL\big[q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})\big] \Big] \\ L_T &= \mathbb{E}_q \Big[KL\big[q(\mathbf{x}_T|\mathbf{x}_0)||p_\theta(\mathbf{x}_T|\mathbf{y})\big] \Big] \end{cases} \qquad (9)
\end{aligned}
$$

To keep notation concise, we consider $\mathbb{E}_{\mathbf{x}_{0:T}\sim q(\mathbf{x}_{0:T})} = \mathbb{E}_q$ in the previous serie of equation. In Equation (9), L_T could be ignored because it doesn't depend on θ. L_0 is modeled by Ho et al., 2020 using a separate neural network. L_t is a KL between 2 Gaussians distribution, so it could be calculated with a closed form.

We observe that the probability distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is actually tractable (Ho et al., 2020):

$$
q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad \text{with} \quad \begin{cases} \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \\ \tilde{\beta}_t &= \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{cases} \qquad (10)
$$

With $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ and $\tilde{\beta}_t\mathbf{I}$ the mean and the variance of $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, respectively. Using Equation (6) we can express $\mathbf{x}_0$ in a convenient way:

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon) \tag{11}$$

We can then simplify $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ in Equation (10):

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\Big) \tag{12}$$

Similarly, we can re-parameterize $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{y})$ because $\mathbf{x}_t$ is available as input at training time:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\Big) \tag{13}$$

We compute L_t (from Equation (9)) by using the closed-form formula of the KL between 2 Gaussian distributions:

$$\begin{aligned}
L_t &= \mathbb{E}_q\left[\frac{1}{2\left\|\sigma_t^2\right\|_2^2}\left\|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\right\|_2^2\right] \\
&= \mathbb{E}_q\left[\frac{1}{2\left\|\sigma_t^2\right\|_2^2}\left\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\Big)\right\|_2^2\right] \quad \text{using Eqs. 12 and 13} \\
&= \mathbb{E}_q\left[\frac{(1-\alpha_t)^2}{2\alpha_t(1-\bar{\alpha}_t)\left\|\sigma_t^2\right\|_2^2}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2\right] \qquad (14)
\end{aligned}$$

One could simplify the loss shown in Equation (14) (Ho et al., 2020):

$$L_t = \mathbb{E}_q\Big[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|_2^2\Big] \tag{15}$$

$$= \mathbb{E}_q\Big[\left\|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\right\|_2^2\Big] \tag{16}$$

F. Details on the DDPM trained on Omniglot

F.1. Architecture

The DDPM and CFGDM models are leveraging a U-Net (Ronneberger et al., 2015) to model ϵ_θ. The U-Net is made of an encoder and a decoder. The architecture is described in detail in Table 3.

Table 3. Description of U-Net architecture of the DDPM and CFGDM

Network	Layer	# params
ConvNext(In$_c$, Out$_c$)	Conv2d(In$_c$, In$_c$, 7, padding=3)	In$_c$ $\times$ In$_c$ $\times$ 7 $\times$ 7 + Inc$_c$
	GroupNorm(In$_c$)	2 x In$_c$
	Conv2d(In$_c$, 3*In$_c$, 3, padding=3)	3*In$_c$ $\times$ In$_c$ $\times$ 3 $\times$ 3 + 3*Inc$_c$
	GeLU	-
	GroupNorm(In$_c$)	6 x Inc$_c$
	Conv2d(3*In$_c$, Out$_c$, 3, padding=3)	3*In$_c$ $\times$ Out$_c$ $\times$ 3 $\times$ 3 + Out$_c$
	## Shortcut connection	
	Conv2d(In$_c$, Out$_c$, 3, padding=3)	In$_c$ $\times$ Out$_c$ $\times$ 3 $\times$ 3 + Out$_c$
TimeEmbedding(In$_c$, Out$_c$)	GeLU	-
	Linear(In$_c$, Out$_c$)	Out$_c$ $\times$ In$_c$
LinearAttention(In$_c$)	Conv2d(In$_c$, 8*In$_c$, 1, padding=0)	In$_c$ $\times$ 8*In$_c$ + 8*In$_c$
	Conv2d(3*In$_c$, In$_c$, 1, padding=0)	3*In$_c$ $\times$ In$_c$ + In$_c$
	GroupNorm(In$_c$)	Inc$_c$
	Conv2d(In$_c$, In$_c$, 4, padding=1)	In$_c$ $\times$ In$_c$ $\times$ 4 $\times$ 4 + In$_c$
DS_U-Net_Block(Inc$_c$, Out$_c$)	ConvNext(Inc$_c$, Out$_c$)	
	TimeEmbedding(192, Out$_c$)	
	ConvNext(Out$_c$, Out$_c$)	
	TimeEmbedding(192, Out$_c$)	
	LinearAttention(Out$_c$)	
	DownSampling(2)	
US_U-Net_Block(Inc$_c$, Out$_c$)	ConvNext(Inc$_c$, Out$_c$)	
	TimeEmbedding(192, Out$_c$)	
	ConvNext(Out$_c$, Out$_c$)	
	TimeEmbedding(192, Out$_c$)	
	LinearAttention(Out$_c$)	
	UpSampling(2)	
U-Net Omniglot	Conv2d(2, 32, 7, padding=3)	2 K
	DS_U-Net_Block(32, 48)	167 K
	DS_U-Net_Block(48, 96)	577 K
	DS_U-Net_Block(96, 192)	1 771 K
	ConvNext(192, 192)	956 K
	LinearAttention(192)	82 K
	ConvNext(192, 192)	956 K
	US_U-Net_Block(2*192, 96)	1 062 K
	US_U-Net_Block(2*96, 48)	297 K
	ConvNext(48, 1)	60K

The encoder of the U-Net is made with 4 layers: a first convolution and 3 down-sampling layers (called DS_U-Net_Block). These down-sampling layers are made with 2 ConvNext layers (Liu et al., 2022) followed by one Linear Attention layer (Li et al., 2020). Each of the feature maps of the ConvNext layer is conditioned with time through the TimeEmbedding Block.

The information bottleneck is composed of 3 layers: a ConvNext layer followed by a Linear attention layer followed by another ConvNext layer.

After the bottleneck, there are 2 up-sampling layers (called US_U-Net_Block). The up-sampling layers are very similar to the down-sampling ones except that they increase the size of the feature maps by a factor of 2. Similarly to the DS_U-Net_Block, each ConvNext layer in the US_U-Net_Block is time-conditioned using the TimeEmbedding layer. In the end, we use a ConvNet layer to equate the number of channels and the size of the output image.

Overall, the base architectures of the **DDPM** and the **CFGDM** on Omniglot have 5.9 million parameters.

F.2. Training details

We schedule the β_t coefficient in Equation (5). β_0 is equal to 1.10^{-4} and β_T to 0.02. The β_T are linearly spanning the time space between 1.10^{-4} and 0.02. In the base architecture T is set to 600

For the training of the parameters of the U-Net model, we use an Adam Optimizer (Kingma & Ba, 2014) with a learning rate of 1.10^{-4}. We train the network for 300 epochs, with a batch size of 128

F.3. Explored hyper-parameters

To obtain the scatter plot in Figure 2a, we have varied certain hyper-parameters:

- The T hyper-parameter, ranging from 200 to 900 with steps of 100 (8 values overall).
- The number of features of the First ConvNext Layer (48 in the base architecture), ranging from 36 to 120 with steps of 12 (8 values overall). Note that this hyper-parameter has a strong impact on the total number of parameters of the U-Net network because the number of features of the subsequent ConvNext blocks depends on the number of features of the first ConvNext layer (it is multiplied by 2 at every layer).

Overall we have plotted the diversity and the accuracy of 64 **DDPM** models in Figure 2a.

F.4. DDPM samples on Omniglot

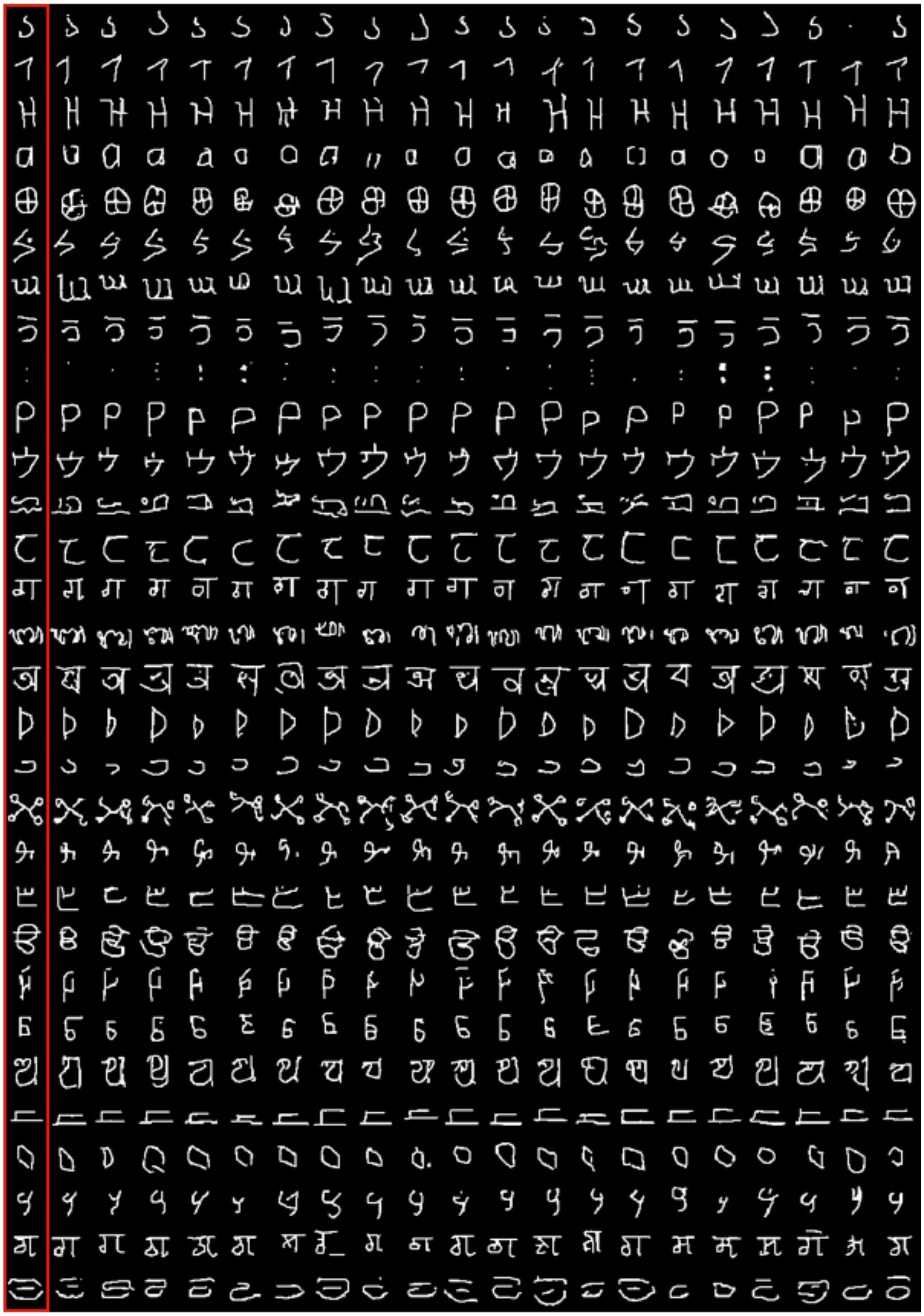


Figure F.1. Samples generated by the DDPM on Omniglot. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 150 concepts) from the Omniglot test set. Each line is composed of 20 DDPM samples that represent the same visual concept.

G. Details on the CFGDM trained on Omniglot

G.1. Loss of the CFGDM

Dhariwal & Nichol, 2021 proposed to improve the conditioning signal of the DDPM using a classifier. To do so, the authors suggest the following form of conditional probability distribution:

$$p_{\theta,\gamma}(\mathbf{x}|\mathbf{y}) \propto p_\theta(\mathbf{x}) \cdot p_\theta(\mathbf{y}|\mathbf{x})^{1+\gamma} \tag{17}$$

In this equation, $p_\theta(\mathbf{y}|\mathbf{x})$ is a classifier (trained separately). The score function of the corresponding diffusion model is then:

$$\nabla_{\mathbf{x}_t} \log p_{\theta,\gamma}(\mathbf{x}_t|\mathbf{y}) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) + (1+\gamma)\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{y}|\mathbf{x}_t) \tag{18}$$

The second term of the RHS of Equation (18) requires to train a classifier ($\log p_\theta(\mathbf{y}|\mathbf{x}_t)$). Training such a classifier is not convenient because it has to be trained to recognize degraded samples (the $\mathbf{x}_t$ are the degraded versions of the original image). To circumvent this issue, Ho & Salimans, 2022 apply the Bayes' rule to replace $p_\theta(\mathbf{y}|\mathbf{x}_t)$:

$$p_\theta(\mathbf{y}|\mathbf{x}_t) = \frac{p_\theta(\mathbf{x}_t|\mathbf{y})p_\theta(\mathbf{y})}{p(\mathbf{x}_t)} \tag{19}$$

Equation (18) now becomes:

$$\nabla_{\mathbf{x}_t} \log p_{\theta,\gamma}(\mathbf{x}_t|\mathbf{y}) = (1+\gamma)\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y}) - \gamma\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t) \tag{20}$$

G.2. Practical considerations for $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$ and $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y})$

The loss in Equation (20) is particularly convenient as one can train a single model to evaluate both $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$ and $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y})$. In the section Appendix E.2, we have shown that $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y})$ could be modeled with an auto-encoder ϵ_θ ($\epsilon_\theta : \mathbb{R}^D \times \mathbb{R}^D \rightarrow R^D$). We can actually use the same auto-encoder, with a non-informative conditioning signal, to model also $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$. In practice, if we want to model $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{y})$, we fed the auto-encoder ϵ_θ with a concatenation of the noisy input image ($\mathbf{x}_t$) and the corresponding exemplars ($\mathbf{y}$). In this case, we use the notation $\epsilon_\theta(\mathbf{x}_t, \mathbf{y})$. To model $\nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$, we fed the network with the noisy image ($\mathbf{x}_t$), concatenated with a black image. In this case, we use the notation $\epsilon_\theta(\mathbf{x}_t, \varnothing)$. In practice, we use a drop-out function, to randomly drop some of the informative exemplars and replace them with a black image (following a Bernoulli distribution). We set the drop-out probability to 0.1.

G.3. Architecture and training

The architecture and the training details of the CFGDM model on the Omniglot dataset are exactly the same as those of the DDPM on the Omniglot dataset (see Appendix F).

G.4. Explored hyper-parameters

To obtain the scatter plot in Figure 2a, we have varied certain hyper-parameters:

- The T hyper-parameter, ranging from 200 to 900 with steps of 100 (8 values overall).
- The number of features of the First ConvNext Layer (48 in the base architecture), ranging from 12 to 96 with steps of 12 (8 values overall). Note that this hyper-parameter has a strong impact on the total number of parameters of the U-Net network because the number of features of the subsequent ConvNext blocks depends on the number of features of the first ConvNext layer (it is multiplied by 2 at every layer).
- The guidance scale with the following values : (0.5, 1, 2, 3, 5). Note, that we do not have to retrain the model to change the guidance scale, as the change is occurring only during sampling.

Overall we have plotted the diversity and the accuracy of 320 CFGDM models in Figure 2a.

G.5. CFGDM samples on Omniglot

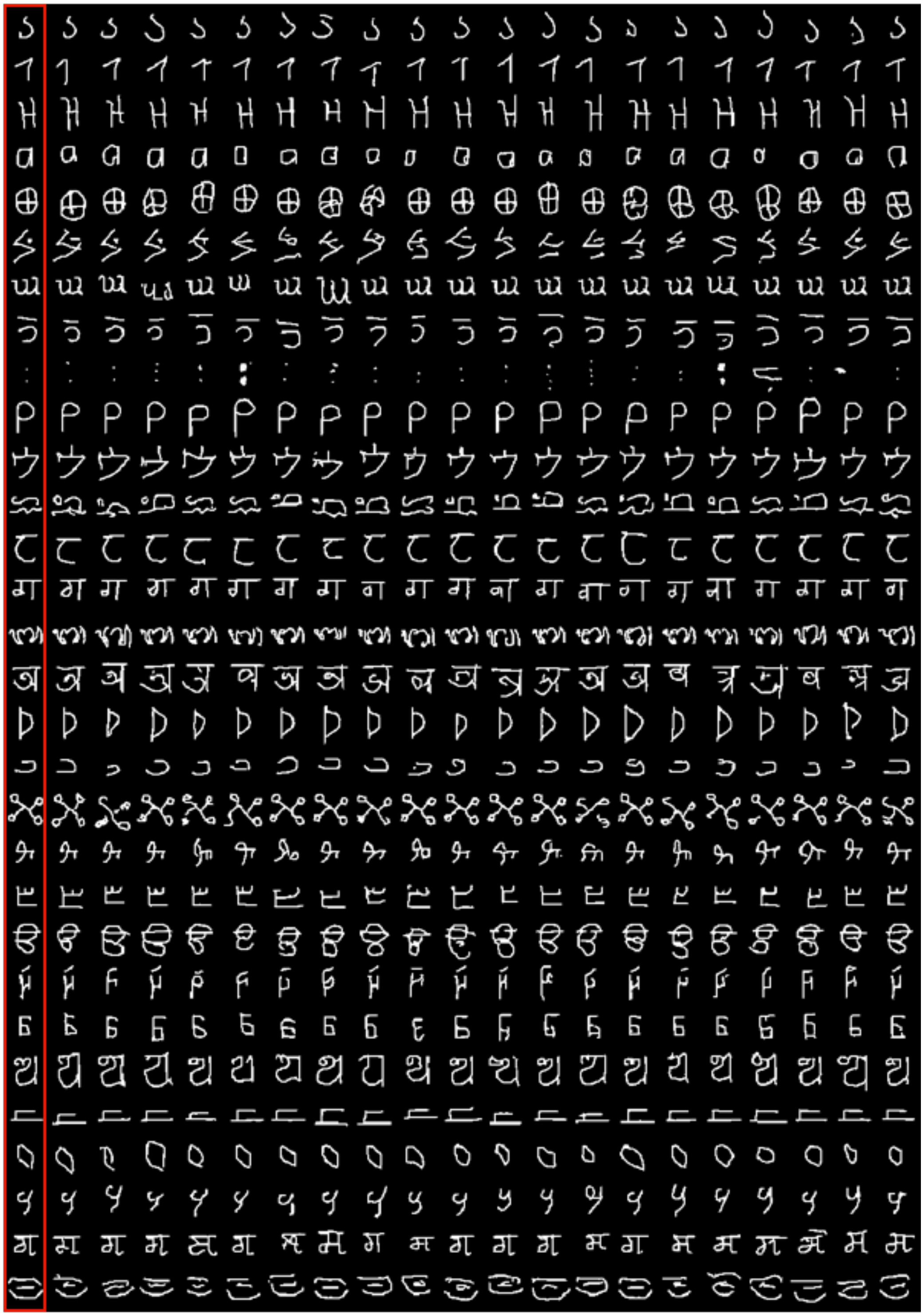


Figure G.1. Samples generated by the CFGDM on Omniglot. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 150 concepts) from the Omniglot test set. Each line is composed of 20 CFGDM samples that represent the same visual concept.

H. Details on the FSDM trained on Omniglot

H.1. Conditioning

The FSDM offers an alternative to condition the DDPM models. Instead of conditioning the U-Net network of the DDPM with a single image, the FSDM proposed to condition it with a context vector that aggregates the information from a context set. When the FSDM is trained on Omniglot we condition it using a mechanism similar to FiLM (Perez et al., 2017). We use a U-Net to extract a context vector from the set of samples presented to the model. The context is in the form of a vector, which is used to condition the intermediate feature maps $\mathbf{u}_t$ in the DDPM U-Net. Note that the feature map $\mathbf{u}_t$ is obtained when the U-Net input is $\mathbf{x}_t$. We can represent the conditioning as:

$$\mathbf{p}_t = m(\mathbf{c})\mathbf{u}_t + b(\mathbf{c}) \tag{21}$$

Here, m and b are learnable and context-dependent neural networks. Additionally, we merge together $\mathbf{c}$ with the time-step embedding, and using that we define a generic per-step conditioning mechanism for each layer.

H.2. Architecture

As the backbone, FSDM utilizes the same architecture as the DDPM trained on Omniglot.

For the context net, as mentioned above we utilize a U-Net. The architecture of the Encoder U-Net is described in detail in Table 4. For the described architecture we consider the number of residual blocks to be 2.

The size of the model is 6.6 million parameters out of which 2.5 million parameters are for the encoder and 4.1 million are for the generative model.

H.3. Training details

The attention resolution, learning rate, optimizer, and batch size we use are the same as that implemented in `https://github.com/georgosgeorgos/few-shot-diffusion-models`. The size of the context channels and hidden dimensions is set to 128.

We train the model for 300 epochs.

H.4. Explored hyper-parameters

To obtain the scatter plot in Figure 2a, we varied the following hyper-parameters:

- The sample size, ranging from 2 to 6 with steps of 1 (5 values overall)
- The number of time-steps or diffusion steps ranging from 200 to 100 with steps of 200 (5 values overall)
- The number of residual blocks per downsample, ranging from 1 to 4 (4 values overall)

We have trained 100 different FSDM models and plotted the diversity and the accuracy in Figure 2a.

Table 4. Description of the Encoder U-Net architecture

Network	Layer	# params
DownSample()	AvgPool2D(kernel_size=2, stride=2)	0
AttnPool2D(Sp$_d$,Emb$_d$,NumHeads$_c$,Out$_d$)	Conv1D(Emb$_d$.3*Emb$_d$,1,1)	49.5 K
	QKVAttention(Emb$_d$ // NumHeads$_c$)	0
	Conv1D(Emb$_d$,Out$_d$,1,1)	16.5 K
ResBlock(In$_c$, Out$_c$, Emb$_c$, down=$False$)	GroupNorm(32,In$_c$)	
	if (down) : DownSample()	0
	if (down) : DownSample()	0
	SiLU	0
	Conv2D(In$_c$,Out$_c$,3,1,1)	
	SiLU	0
	Linear(Emb$_c$, 2× Out$_c$)	
	GroupNorm(32,Out$_c$)	
	SiLU	0
	Dropout	0
	Conv2D(Out$_c$,Out$_c$,3,1,1)	
	if (In$_c$!= Out$_c$) : Conv2D(In$_c$,Out$_c$,1,1)	
AttnBlock(In$_c$, NumHeads, NumHeads$_c$)	GroupNorm(32, In$_c$)	
	Conv1D(In$_c$,3× In$_c$,1,1)	
	QKVAttentionLegacy(In$_c$ // NumHeads$_c$)	0
	Conv1D(In$_c$,In$_c$,1,1)	
InputBlock()	Conv2D(1,In$_c$,3,1,1)	320
	ResBlock(In$_c$=32,Out$_c$=32, Emb$_c$=128)	26.9 K
	ResBlock(In$_c$=32,Out$_c$=32, Emb$_c$=128)	26.9 K
	ResBlock(In$_c$=32,Out$_c$=32, Emb$_c$=128, down=$True$)	26.9 K
	ResBlock(In$_c$=32,Out$_c$=64, Emb$_c$=128)	74.2 K
	ResBlock(In$_c$=64,Out$_c$=64, Emb$_c$=128)	90.6 K
	ResBlock(In$_c$=64,Out$_c$=64, Emb$_c$=128, down=$True$)	90.6 K
	ResBlock(In$_c$=64,Out$_c$=96, Emb$_c$=128)	169.8 K
	ResBlock(In$_c$=96,Out$_c$=96, Emb$_c$=128)	191.2 K
	ResBlock(In$_c$=96,Out$_c$=96, Emb$_c$=128, down=$True$)	191.2 K
	ResBlock(In$_c$=96,Out$_c$=128, Emb$_c$=128)	304.2 K
	AttnBlock(In$_c$=128, NumHeads=1, NumHeads$_c$=64)	66.3 K
	ResBlock(In$_c$=128,Out$_c$=128, Emb$_c$=128)	328.7 K
	AttnBlock(In$_c$=128, NumHeads=1, NumHeads$_c$=64)	66.3 K
MiddleBlock()	ResBlock(In$_c$=128,Out$_c$=In$_c$, Emb$_c$=128)	328.7 K
	AttnBlock(In$_c$=128, NumHeads=1, NumHeads$_c$=64)	66.3 K
	ResBlock(In$_c$=128,Out$_c$=In$_c$, Emb$_c$=128)	328.7 K
Encoder U-Net	Linear(32, 128)	4.2 K
	SiLU	0
	Linear(128, 128)	16.5 K
	InputBlock()	1653.8 K
	MiddleBLock()	723.7 K
	GroupNorm(32,128)	256
	SiLU	0
	AttnPool2D(Sp$_d$=6,Emb$_d$=128,NumHeads$_c$=64,Out$_d$=128)	66.0 K

Note : Blue layers represent variable layers dependent on a certain parameter

H.5. FSDM samples on Omniglot

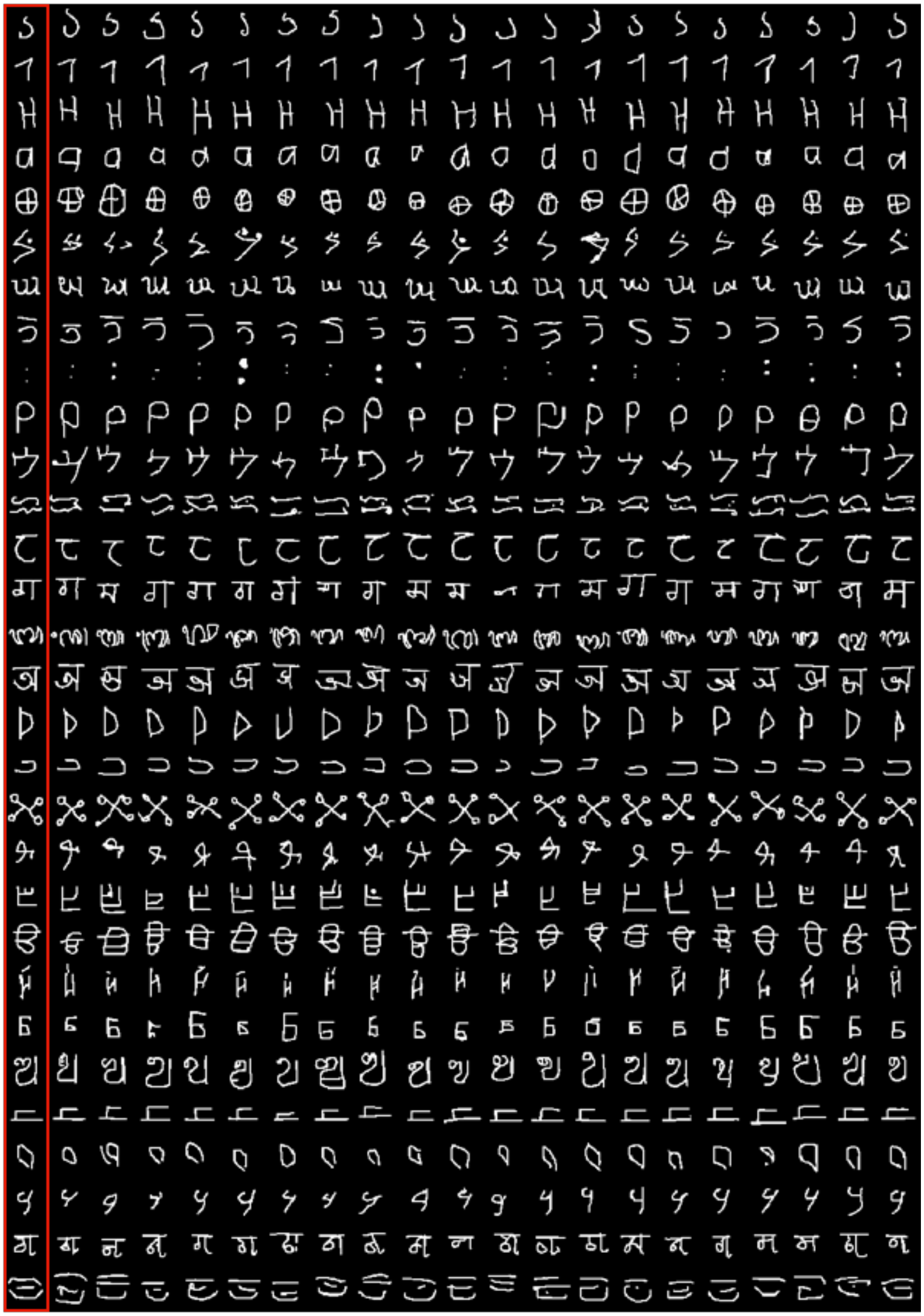


Figure H.1. Samples generated by the FSDM on Omniglot. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 150 concepts) from the Omniglot test set. Each line is composed of 20 FSDM samples that represent the same visual concept.

I. Details on the DDPM trained on QuickDraw-FS

I.1. Architecture and training

The DDPM trained on QuickDraw-FS has a similar architecture to the DDPM trained on Omniglot (see Appendix F). The only difference is that the base architecture we use on QuickDraw-FS has 60 channels in the first ConvNext block. The total number of parameters of the base architecture on QuickDraw-FS is 13,1 million.

All the training hyper-parameters (batch size, learning rate, beta scheduling) are the same as the DDPM on Omniglot.

I.2. Explored hyper-parameters

To obtain the scatter plot in Figure 2b, we have varied certain hyper-parameters:

- The T hyper-parameter, ranging from 200 to 900 with steps of 100 (8 values overall).
- The number of features of the First ConvNext Layer (48 in the base architecture), ranging from 24 to 108 with steps of 12 (8 values overall). Note that this hyper-parameter has a strong impact on the total number of parameters of the U-Net network because the number of features of the subsequent ConvNext blocks depends on the number of features of the first ConvNext layer (it is multiplied by 2 at every layer).

We have repeated all this hyper-parameters exploration for 2 different random seeds (i.e., different weight initialization). Overall we have plotted the diversity and the accuracy of 128 models in Figure 2b.

I.3. DDPM samples on QuickDraw-FS

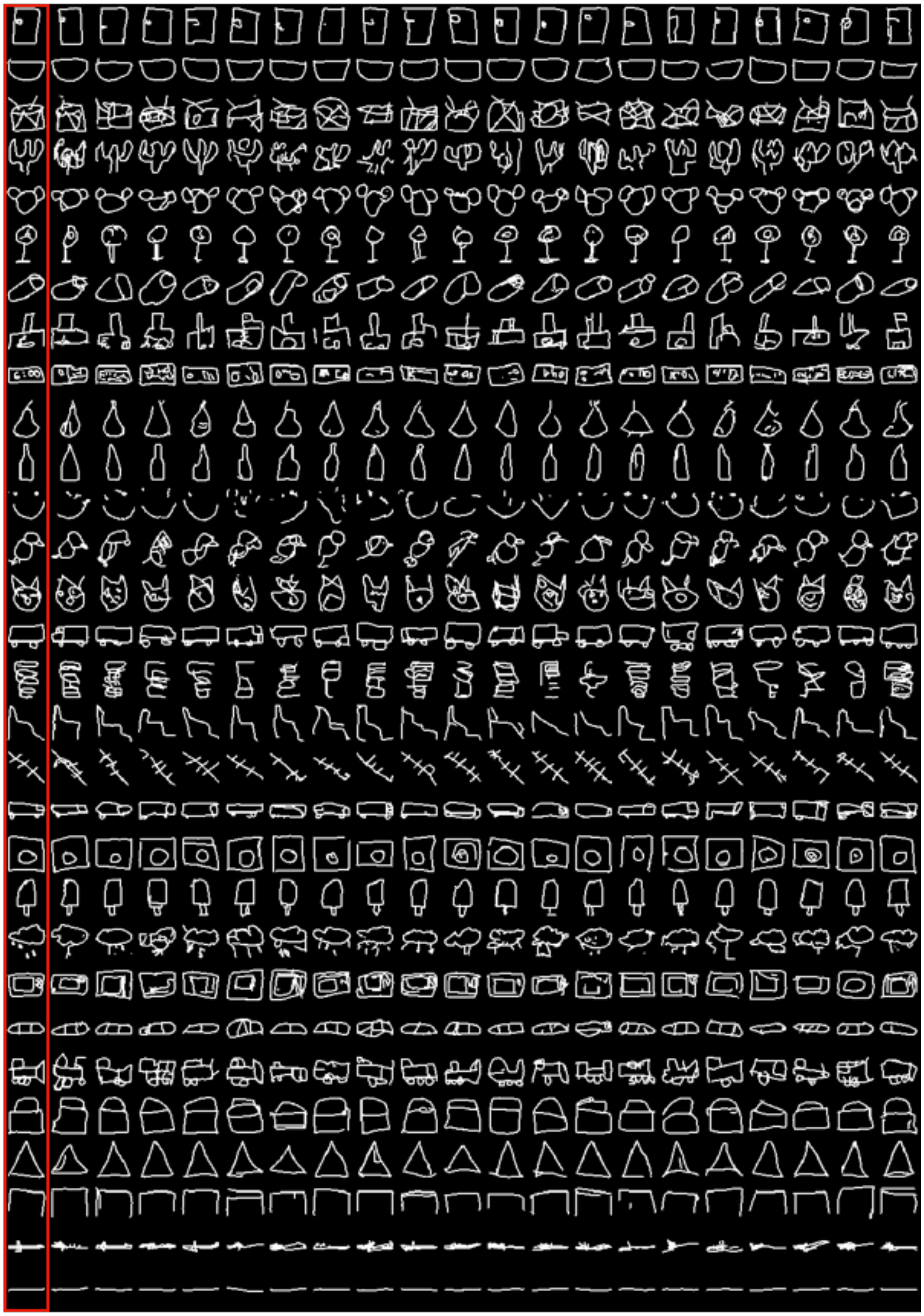


Figure I.1. Samples generated by the DDPM on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 DDPM samples that represent the same visual concept.

J. Details on the CFGDM trained on QuickDraw-FS

J.1. Architecture and training

The base architecture and the training details of the CFGDM on QuickDraw-FS are exactly the same as those of the DDPM on the QuickDraw-FS dataset (see Appendix I).

J.2. Explored hyper-parameters

To obtain the scatter plot in Figure 2b, we have varied certain hyper-parameters:

- The T hyper-parameters, ranging from 200 to 900 with steps of 100 (8 values overall).
- The number of features of the First ConvNext Layer (60 in the base architecture), ranging from 24 to 108 with steps of 12 (8 values overall). Note that this hyper-parameter has a strong impact on the total number of parameters of the U-Net network, as the number of features of the subsequent ConvNext block is equal to the previous number of features multiplied by 2...
- The guidance scale with the following values : $(0.5, 1, 2, 3, 5)$. Note, that we do not have to retrain the model to change the guidance scale, as the change is occurring only during sampling.

We have repeated all this hyper-parameters exploration for 2 different random seeds (i.e., different weight initialization). Overall we have plotted the diversity and the accuracy of 320 models in Figure 2b.

J.3. CFGDM samples on QuickDraw-FS

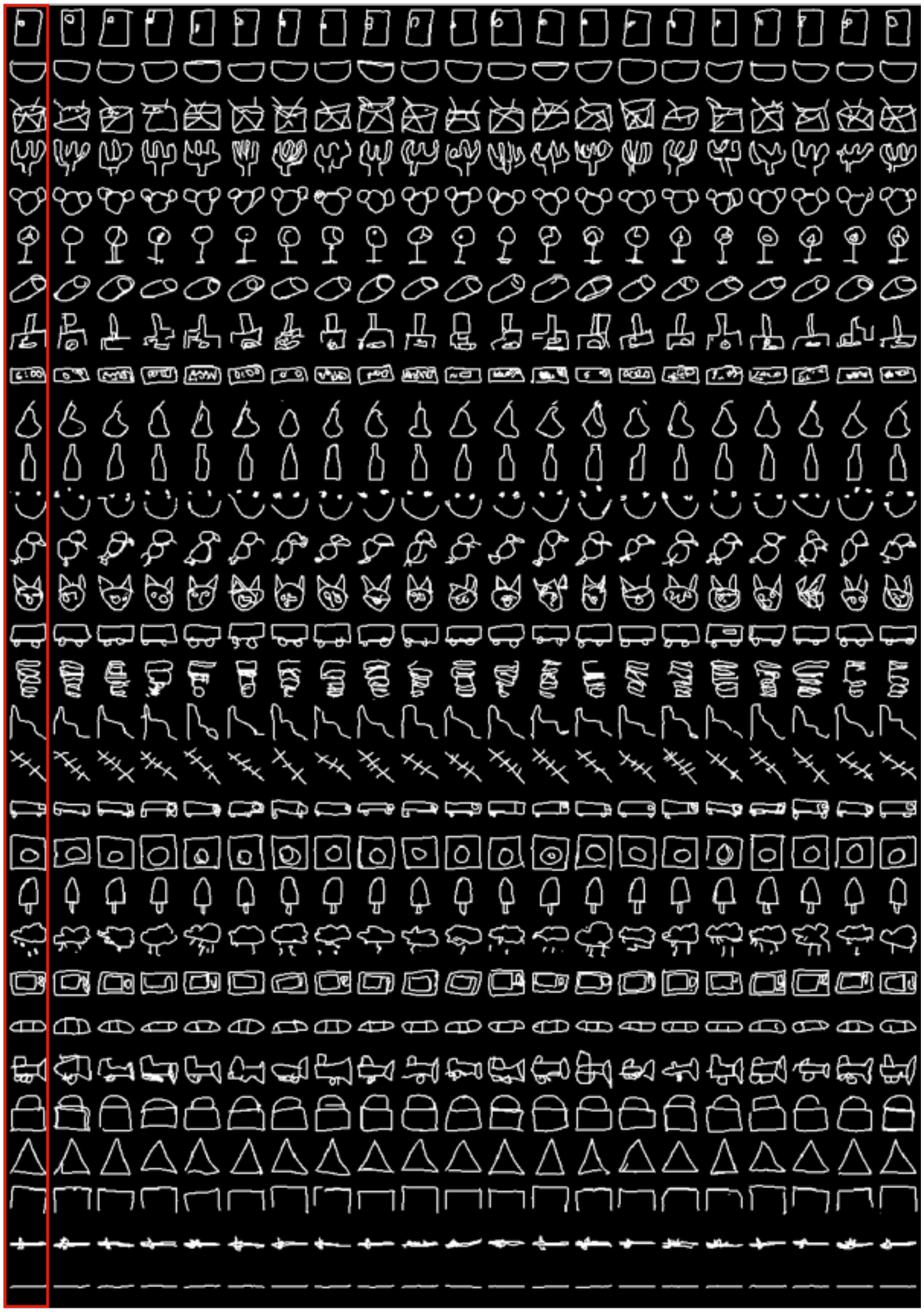


Figure J.1. Samples generated by the CFGDM on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 CFGDM samples that represent the same visual concept.

K. Details on the VAE-NS trained on QuickDraw-FS

K.1. Architecture

For the VAE-NS on QuickDraw-FS we use a similar architecture to the VAE-NS on Omniglot (described in S9 of Boutin et al., 2022). Here, we remind the reader of the main properties of the VAE-NS network.

The VAE-NS network is composed of different sub-networks:

- Shared encoder $x \mapsto h$: An instance encoder E that takes each individual datapoint x_i to a feature representation $h_i = E(x_i)$.
- Statistic network $q(c|D, \phi) : h_1, ..., h_k \mapsto \mu_c, {\sigma^2}_c$: A pooling layer that aggregates the matrix $(h_1, ..., h_k)$ to a single pre-statistic vector v. Edwards & Storkey, 2016 uses sample mean for their experiments. Which is followed by a post-pooling network that takes v to a parametrization of a Gaussian.
- Inference network $q(z|x, c, \phi) : h, c \mapsto \mu_z, {\sigma^2}_z$: Inference network gives an approximate posterior over latent variables.
- Latent decoder network $p(z|c; \theta) : c \mapsto \mu_z, {\sigma^2}_z$
- Observation decoder network $p(x|c, z; \theta) : c, z \mapsto \mu_x$

Compared to the Omniglot version, we have increased the number of parameters to fit the higher complexity of the QuickDraw-FS dataset. In particular, we have increased the number of stochastic layers in the inference network and the observation decoder network from 1 to 6. The number of stochastic layers controls the number of hierarchical latent variables we use in the encoder and the decoder. To keep the number of parameters reasonable we have decreased the dimension of the context vector, i.e., the output size of the static network, from 512 to 128. In practice, we have found that reducing this dimension has little impact on the performance while reducing drastically the number of parameters. We have set the dimension of the latent variable to 256. For the base architecture, the size of the last latent variable is set to 80, the number of context samples is equal to 5, and the number of layers (per stochastic layer) is set to 6.

The base architecture of the VAE-NS has 12.4 million parameters.

K.2. Training details

We have trained the VAE-NS for 300 epochs, using a batch size of 32. We use the Adam optimizer to update the weights of the network, with a learning rate of 1.10^{-3}.

K.3. Explored hyper-parameters

To obtain the scatter plot in Figure 2b, we have varied certain hyper-parameters:

- The number of dimensions of the last latent variable, from 40 to 120, by steps of 40 (so 3 different values overall)
- the number of sub-layers that composed each stochastic layer, from 2 to 10 with step 4 (3 values overall.
- the β coefficient with values: (0.1, 0.5, 1, 2). We remind the reader that the β coefficient in the VAE is used to increase (or decrease if $\beta \leq 1$) to weight of the KL in the ELBO loss (Higgins et al., 2016).
- the number of context samples with values (2, 5, 10). In the VAE-NS, the context samples are used to evaluate the statistics of a specific category (through the statistic network).

Overall we have plotted the diversity and the accuracy of 108 models in Figure 2b.

K.4. VAE-NS samples on QuickDraw-FS

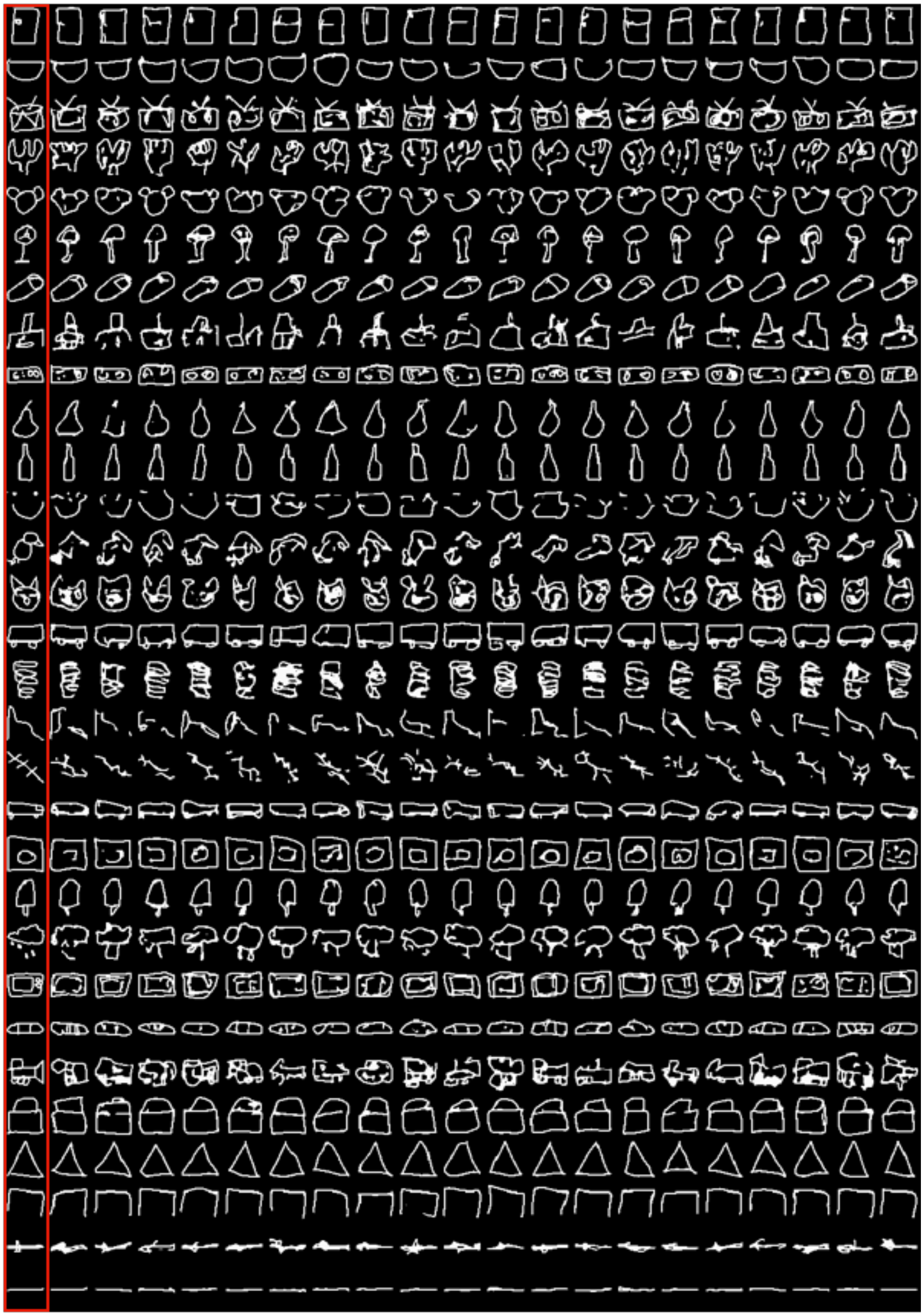


Figure K.1. Samples generated by the VAE-NS on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 VAE-NS samples that represent the same visual concept.

L. Details on the VAE-STN trained on QuickDraw-FS

The VAE-STN is a sequential VAE that allows for the iterative construction of a complex image (Rezende et al., 2016). At each iteration, the algorithm focuses its attention on a specific part of the image ($\boldsymbol{x}$), the prototype ($\tilde{\boldsymbol{x}}$), and the residual image ($\hat{\boldsymbol{x}}$) using the Reading Spatial Transformer Network (STN$_r$). Then the extracted patch is passed to an encoding network (EncBlock) to transform it into a latent variable. This latent variable is concatenated to a patch extracted from the prototype and then passed to the RecBlock network. The produced hidden state is first passed to DecBlock to recover the original patch, and then to the STN$_w$ to replace and rescale the patch into the original image. The LocNet network is used to learn the parameter of the affine transformation we used in the STN. Note that the affine parameters used in STN$_w$ are simply the inverse of those used in STN$_r$.

The STN modules take 2 variables in input: an image (or a patch in the case to the STN$_w$) and a matrix (3×2) describing the parameters of the affine transformation to apply to the input image (Jaderberg et al., 2015). All other modules are made with MLPs networks and are described in Table 5. In the Table 5 we use the following notations:

- s_z: This is the size of the latent space. In the base architecture, we set $s_z = 80$.
- s_{LSTM}: This is the size of the output of the Long-Short Term Memory (LSTM) unit. In the base architecture, we set $s_{LSTM} = 400$
- s_r: This is the resolution of the patches extracted by the Spatial Transformer Net (STN) during the reading operation. In the base architecture, we set $s_r = 15$.
- s_{loc}: This is the number of neurons used at the input of the localization network. In the base architecture, we set $s_{loc} = 150$
- s_w: This is the resolution of the patch passed to the STN network for the writing operation. In the base architecture $s_w = 12$.

For the base architecture, we used $N_{steps} = 80$. The base architecture of the VAE-STN has 11.9 million parameters. For more details on the loss function, please refer to (Rezende et al., 2016).

All other training details are similar to the version trained on Omniglot (see Section S8 in the supplementary information of Boutin et al., 2022).

Table 5. Description of the VAE-STN architecture

Network	Layer	# params
EncBlock(s_r, s_{LSTM}, s_z)	Linear($3 \times s_r^2 + s_{LSTM}$, 2048)	$(3 \times s_r^2 + s_{LSTM}) \times 2048)$ $+ 2048$
	ReLU	
	Linear(2048, 1024)	2097 K
	ReLU	
	Linear(1024, 1024)	1050 K
	ReLU	
	Linear(1024, 512)	524 K
	ReLU	-
	Linear(512, 128)	65 K
	ReLU	-
	Linear(128, $2\times$ s_z)	$256\times s_z + 2\times s_z$
LocNet(s_{loc})	Linear(s_{loc}, 64)	$s_{loc} \times 64 + 64$
	ReLU	-
	Linear(64, 32)	2 K
	ReLU	-
	Linear(32, 6)	0.2 K
DecBlock(s_{LSTM}, s_{loc}, s_w)	Linear(s_{LSTM} - s_{loc}, 2048)	$(s_{LSTM} - s_{loc})\times 2048 + 2048$
	ReLU	-
	Linear(2048, 1024)	2097 K
	ReLU	-
	Linear(1024, 512)	525 K
	ReLU	-
	Linear(512, 256)	131 K
	ReLU	-
	Linear(256, $4 * s_w^2$)	$256 \times 4\times s_w^2 + 4 * s_w^2$
RecBlock(s_z, s_r, s_{LSTM})	LSTMCell($s_z + s_r^2$, s_{LSTM})	$4 \times \big(s_z + s_r^2)\times s_{LSTM}$ $+ s_{LSTM}^2 + s_{LSTM}\big)$
VAE-STN	EncBlock(15 , 400, 80)	$5,4$ K
	RecBlock(80, 12, 400)	$2,998$ K
	DecBlock(400, 150, 15)	$3,552$ K
	LocNet(150)	$11.9K$

L.1. VAE-STN samples on QuickDraw-FS

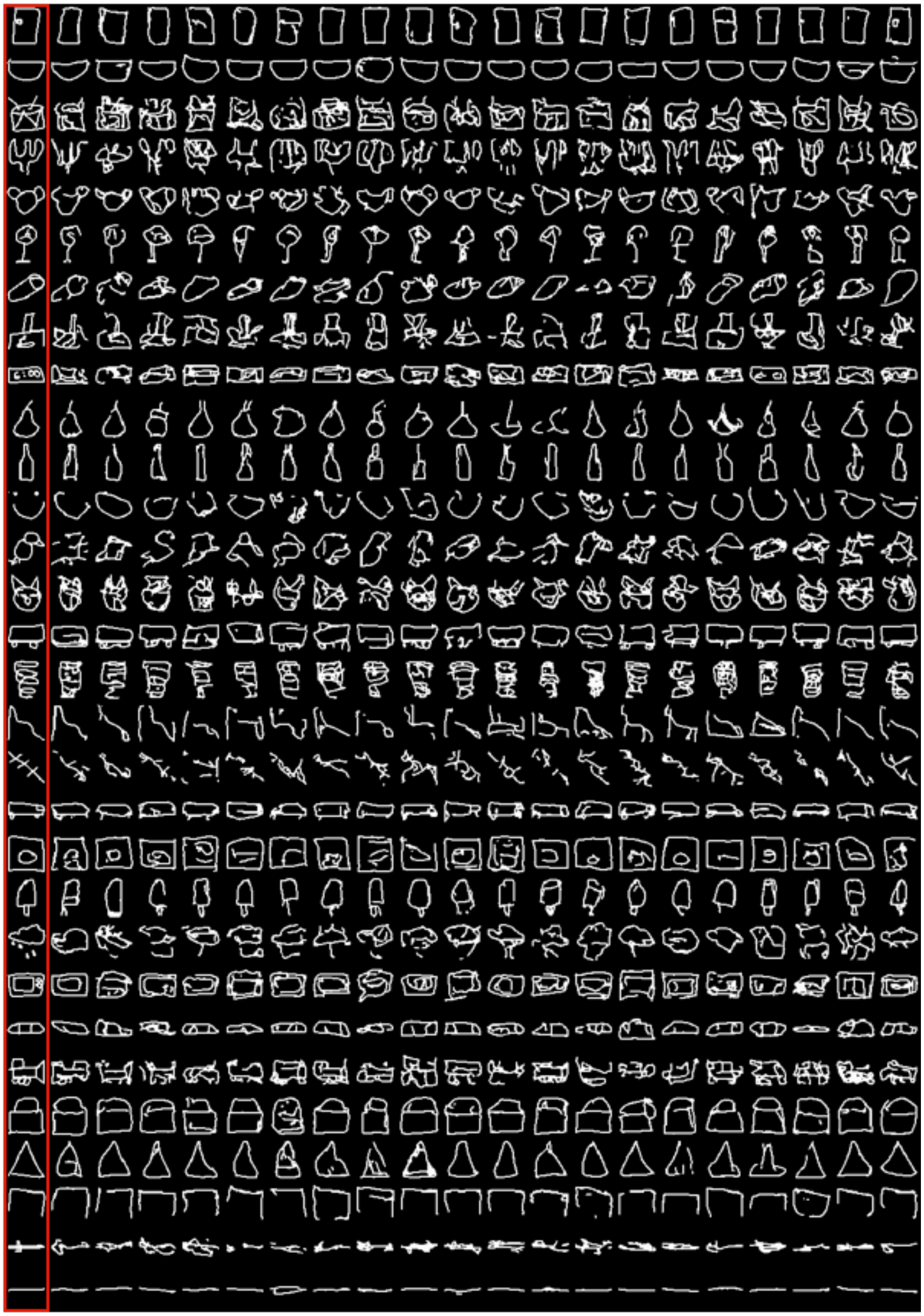


Figure L.1. Samples generated by the VAE-STN on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 VAE-STN samples that represent the same visual concept.

M. Details on the DA-GAN-UN and DA-GAN-RN trained on QuickDraw-FS

M.1. Architecture

The DA-GAN architecture used for the Omniglot dataset is derived from Boutin et al., 2022. For QuickDraw-FS dataset, the same architecture has been extended as shown in Table 6. DA-GAN-UN and DA-GAN-RN model refers to the version whose generator is based on the U-Net and ResNet architecture respectively. Therefore, the difference between the two models is due to the presence of skip connections in DA-GAN-UN model. Following are the details on the DA-GAN's generator model:

- s_z: It represents the size of the latent space. In the base architecture, we have used s_z = 120.
- $G(x, z)$: This is the generator of DAGAN model, which takes an exemplar x and gaussian noise z as input to generate new samples.

The base architecture of DAGAN's generator has 10.5 million parameters. All the DAGAN training details are similar to its Omniglot version (refer to Sections S10 and S11 in the supplementary section of Boutin et al., 2022.

Table 6. Description of the Data Augmentation GAN Architecture

Network	Layer	# params

ConvBlock(In$_c$, Out$_c$, s$_l$)	Conv2d(In$_c$, Out$_c$, 3, stride=s$_l$, padding=1)
	LeakyReLU(0.2), BatchNorm2d(Out$_c$)
		Out$_c$ $\times$ (In$_c$ $\times$ 3 $\times$ 3 + 1)
		2 x Out$_c$

DeConvBlock(In$_c$, Out$_c$, s$_l$)	ConvTranspose2d(In$_c$, Out$_c$, 3, stride=s$_l$, padding=1)
	LeakyReLU(0.2), BatchNorm2d(Out$_c$)
		Out$_c$ $\times$ (In$_c$ $\times$ 3 $\times$ 3 + 1)
		2 x Out$_c$

EncoderBlock(In$_p$, In$_c$, Out$_c$)	ConvBlock(In$_p$, In$_p$)
	ConvBlock(In$_c$ + In$_p$, Out$_c$)
	Conv2d(In$_c$+ Out$_c$, Out$_c$)
	Conv2d(In$_c$+ 2 $\times$ Out$_c$, Out$_c$)
	Conv2d(In$_c$+ 3 $\times$ Out$_c$, Out$_c$)	

DecoderBlock(In$_p$, In$_c$, Out$_c$)	DeConvBlock(In$_p$, In$_p$, 1)
	ConvBlock(In$_c$+In$_p$, Out$_c$, 1)
	DeConvBlock(In$_p$, In$_p$, 1)
	ConvBlock(In$_c$ + In$_p$ + Out$_c$, Out$_c$, 1)
	DeConvBlock(In$_p$, In$_p$, 1)
	ConvBlock(In$_c$ + In$_p$ + 2$\times$ Out$_c$, Out$_c$, 1)
	DeConvBlock(In$_c$ + 2 $\times$ Out$_c$, Out$_c$, 1)	

Generator(s$_z$)	ConvBlock(1, 64, 2)
	EncoderBlock(1, 64, 64)
	EncoderBlock(64, 64, 128)
	EncoderBlock(128, 128, 128)
	Linear(s$_z$, 4$\times$4$\times$8)
	DecoderBlock(0, 136, 64)
	Linear(s$_z$, 7$\times$7$\times$4)
	DecoderBlock(128, 260, 64)
	Linear(s$_z$, 13$\times$13$\times$2)
	DecoderBlock(128, 194, 64)
	DecoderBlock(64, 128, 64)
	DecoderBlock(64, 65, 64)
	ConvBlock(64, 64, 1)
	ConvBlock(64, 64, 1)
	Conv2d(64, 1, 3, stride=1, padding=1)
		10,567,811

M.2. DA-GAN-RN samples on QuickDraw-FS

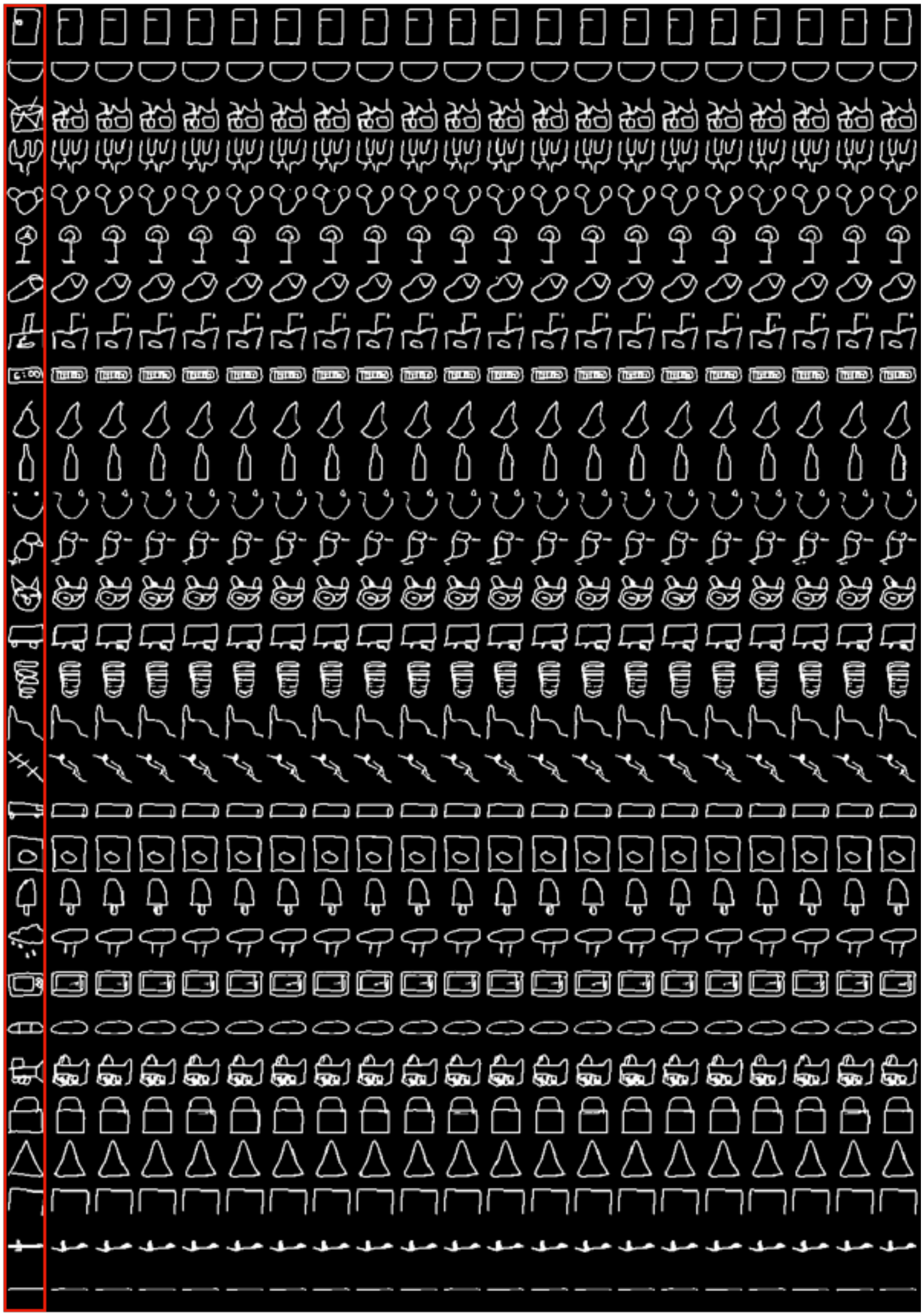


Figure M.1. Samples generated by the DA-GAN-RN on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 DA-GAN-RN samples that represent the same visual concept.

M.3. DA-GAN-UN samples on QuickDraw-FS

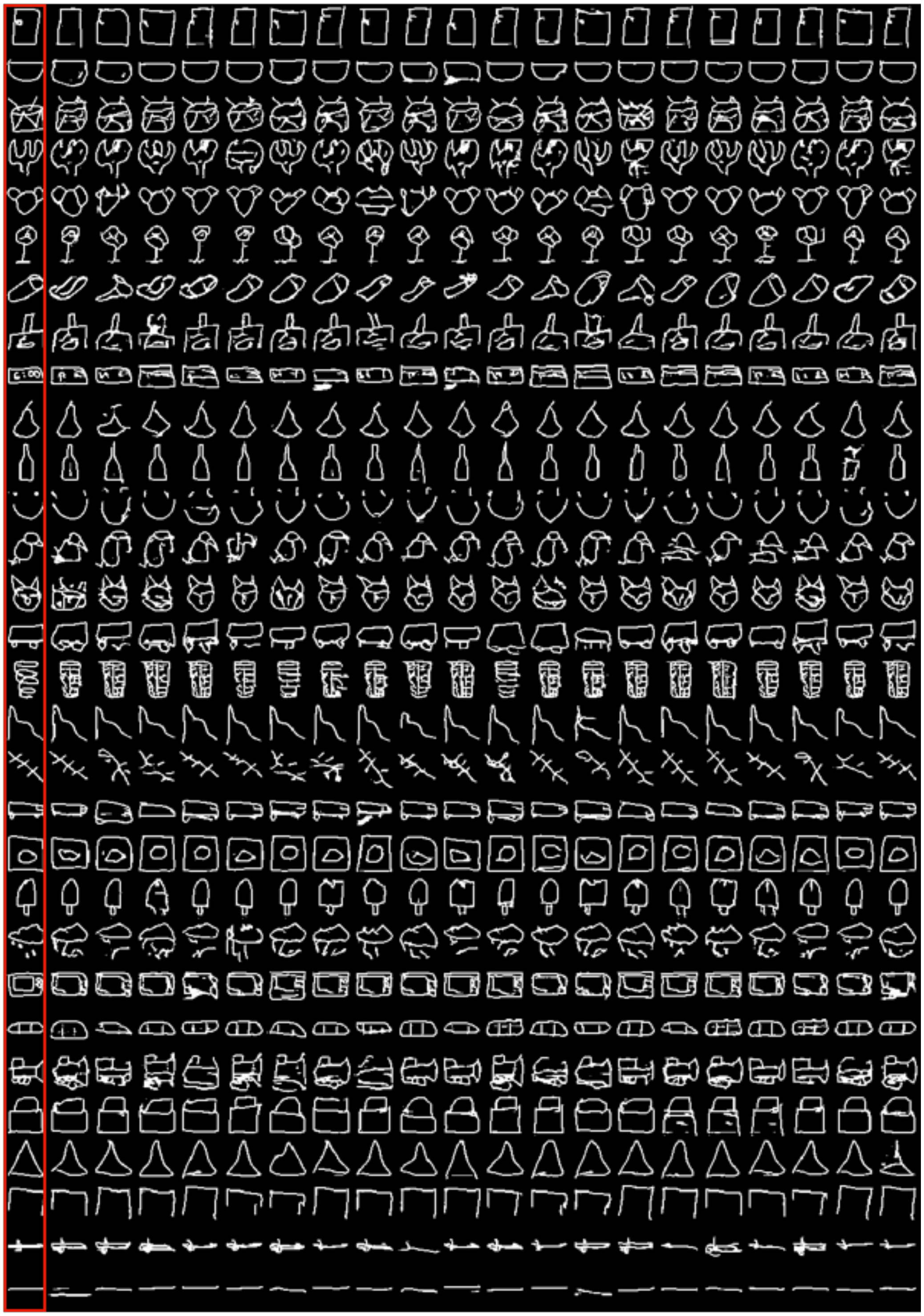


Figure M.2. Samples generated by the DA-GAN-UN on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 DA-GAN-UN samples that represent the same visual concept.

N. Details on the FSDM trained on QuickDraw-FS

N.1. Conditioning

Unlike FSDM trained on Omniglot, for QuickDraw-FS the context net we use to extract context from the set of samples is an sViT similar to Lee et al., 2021. Furthermore, unlike FSDM trained on Omniglot, the context is not calculated on the entire set of images, but rather we utilize a per-patch aggregation, wherein we take a mean value of each patch over the entire set and use that to generate the context vector. We can process any sample size using per-patch aggregation without increasing the number of tokens needed to condition the U-Net, and more crucially, we are able to composite information from various different samples simultaneously.

The conditioning mechanism to combine the context with the generator U-Net is the same as that in FSDM trained on Omniglot (see Appendix H).

N.2. Architecture

As mentioned in Appendix H, for the backbone, FSDM utilizes the same architecture as the DDPM.

The context net utilized for QuickDraw-FS as mentioned above is an sViT similar to Lee et al., 2021 where we handle small sets (1-10) of images. The architecture is described in Table 7.

The size of the model is 12.9 million parameters out of which 5.7 million parameters are for the encoder and 7.2 million are for the generative model.

N.3. Training details

The attention resolution, learning rate, optimizer, and batch size we use are the same as that implemented in `https://github.com/georgosgeorgos/few-shot-diffusion-models`. The size of the context channels and hidden dimensions is set to 256.

We train the model for 200 epochs.

N.4. Explored hyper-parameters

To obtain the scatter plot in Figure 2a, we varied the following hyper-parameters:

- The sample size, ranging from 3 to 6 with steps of 1 (4 values overall)
- The number of timesteps or diffusion steps ranging from 200 to 100 with steps of 200 (5 values overall)
- The number of residual blocks per downsample, ranging from 3 to 6 (4 values overall)

We have trained 80 different FSDM models and plotted the diversity and the accuracy in Figure 2a.

Table 7. Description of the sViT architecture

Network	Layer	# params
SPT(patch_dim, dim)	LayernNorm(patch_dim)	
	Linear(patch_dim, dim)	
PreNorm(dim, fn)	LayerNorm(dim)	
	fn	
FeedForward(dim, hidden_dim)	Linear(dim, hidden_dim)	
	GeLU	0
	Dropout()	0
	Linear(hidden_dim, dim)	
	Dropout()	0
LSA(dim, heads, dim_head)	Softmax(dim=-1)	0
	Dropout()	0
	Linear(dim, dim_head × heads ×3, bias = $False$)	
	Linear(dim_head × heads, dim)	
	Dropout()	0
sViT	Linear(128,256)	33.0 K
	SPT(patch_dim = 320, dim=256)	82.8 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	PreNorm(dim = 256, fn = LSA(dim = 256, heads = 12, dim_head = 64)	787.2 K
	PreNorm(dim = 256, fn = FeedForward(dim = 256, hidden_dim = 256)	132.1 K
	mean(dim=1)	0
	LayerNorm(256)	512
	Linear(256,256)	65.8 K

Note : Blue layers represent variable layers dependent on a certain parameter

N.5. FSDM samples on QuickDraw-FS

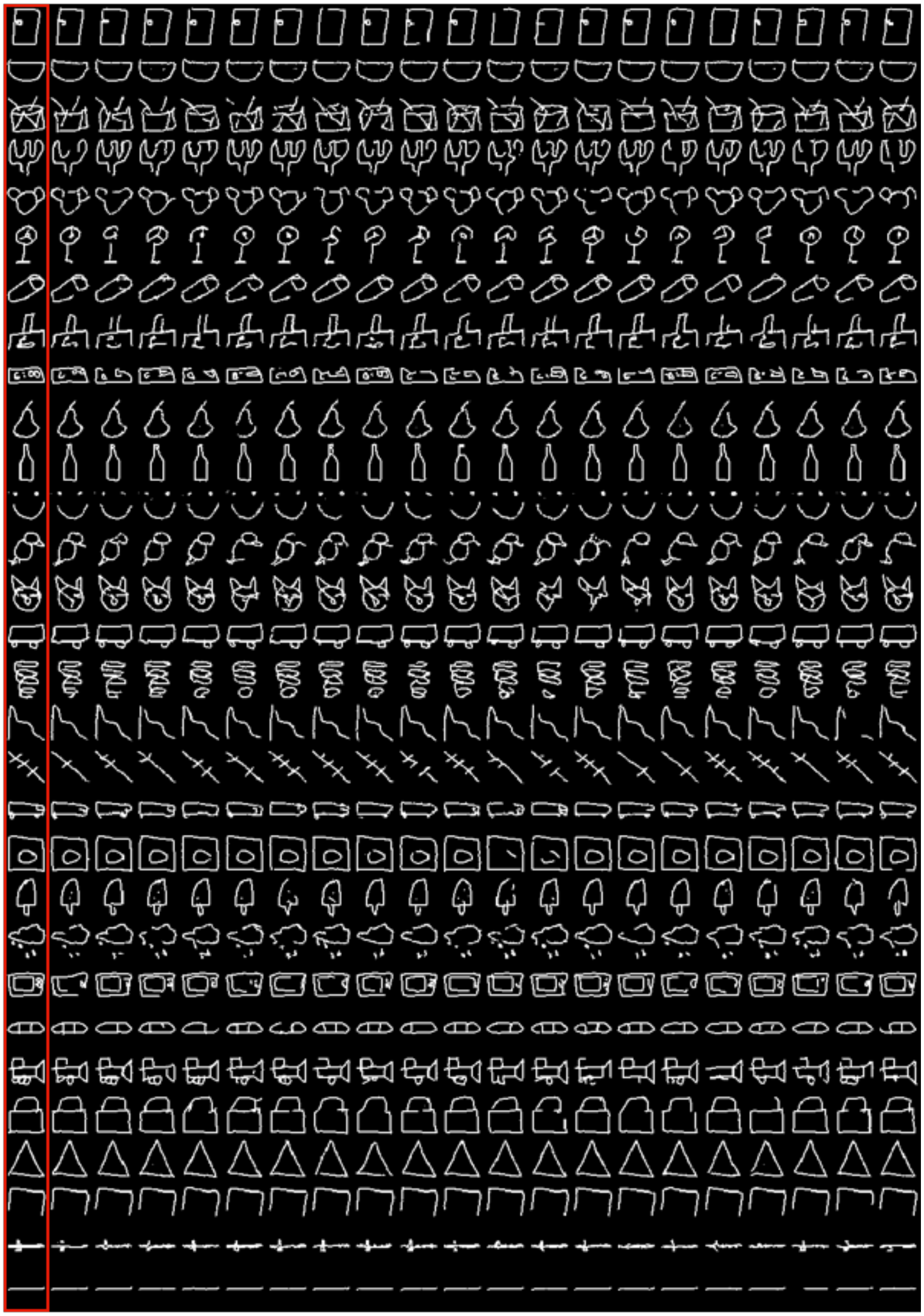


Figure N.1. Samples generated by the FSDM on QuickDraw-FS. All the exemplars used to condition the generative model are in the red frame. The 30 concepts have been randomly sampled (out of 115 concepts) from the QuickDraw-FS test set. Each line is composed of 20 FSDM samples that represent the same visual concept.

O. More details on the *originality* metric

To compute the *originality*, we use the ℓ_2 distance, in the SimCLR latent space, between the exemplar and the samples. We validate the originality metric by comparing it with alternative metrics in which we vary the feature extractor network and the distance metric. As an alternative to the SimCLR network, we consider the Prototypical Net (Snell et al., 2017). For the metric used to compute the distance between the samples and their corresponding exemplars, we have considered the cosine distance. For a given category j, composed with samples v_i^j and exemplars e^j we define the cosine distance in Equation (22). In this equation, f denotes the feature extractor network.

$$d(v_i^j, e^j) = \sqrt{2 - 2C(f(v_i^j), f(e^j))} \quad \text{s.t.} \quad \text{and} \quad C(u, v) = \frac{u \cdot v}{\|u\| \|v\|} \tag{22}$$

In Table 8, we have computed the Spearman rank-order correlation between all possible combinations of distance metrics and feature extractor networks: We observe that all combinations of feature extractor network + metrics have a strong correlation with each other ($\rho > 0.5$) and this correlation is statistically significant ($p < 1 \times 10^{-3}$). It suggests that the way we have defined distance to exemplar is compatible with the other definition. We prefer the SimCLR over the Prototypical feature extractor, because this is a fully unsupervised network (Chen et al., 2020), so it is more convenient to train (no need for labels). Similarly, we choose the ℓ_2-norm to compute the distance between exemplars and samples because it is more natural. In Figure O.1, we show some visual concepts, sorted by originality (in ascending order).

Table 8. Spearman rank-order correlation for different settings

Setting 1	Setting 2	Spearman correlation	p-value
Proto. Net + ℓ_2 distance	Proto. Net + cosine distance	0.97	$4.23. \times 10^{-41}$
Proto. Net + ℓ_2 distance	SimCLR + ℓ_2 distance	0.62	1.14×10^{-34}
SimCLR + ℓ_2 distance	SimCLR + cosine distance	0.85	8.2×10^{-12}
Proto. Net + cosine	SimCLR + cosine distance	0.66	1.02×10^{-21}

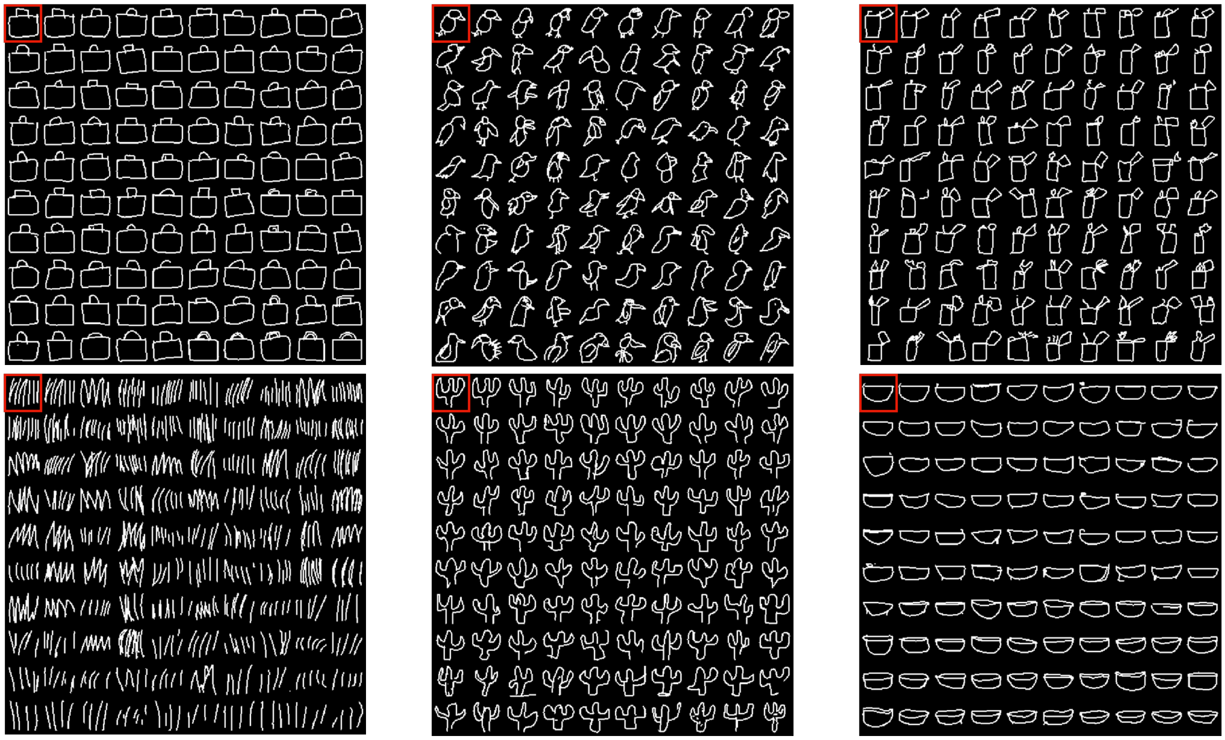


Figure O.1. Randomly picked samples of the QuickDraw-FS test set. The samples are sorted (ascending order) according to their distance from the exemplar using the *originality* metric (SimCLR feature extractor + ℓ_2-norm). The exemplars are highlighted with a red square.

P. Link between originality and diversity

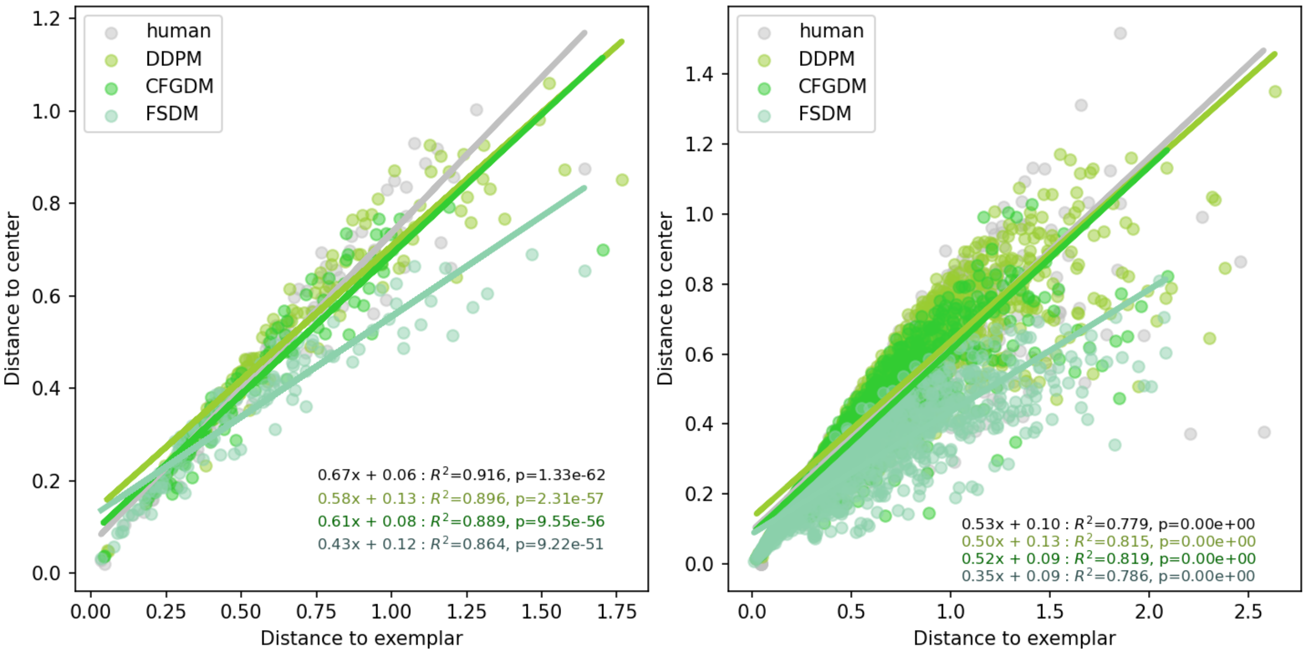
We have defined diversity as the intra-class variability, computed with a standard deviation in the SimCLR feature space. By definition, the intra-class standard deviation is the square root of the mean squared distance between the center of the samples and the samples themselves. For a given category j, composed of N samples v_i^j and a feature space f, the diversity σ_j is defined as in Equation (23).

$$\sigma_j = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\Big(f(v_i^j) - \frac{1}{N}\sum_{i=1}^{N} f(v_i^j)\Big)^2} \tag{23}$$

For a given sample, we have also defined the *originality*. The originality is the ℓ_2 distance, in the SimCLR feature space, between a sample and the corresponding category exemplar. In Equation (24), we write down the formula of average category originality c_j, for a category j represented by an exemplar e^j:

$$c_j = \frac{1}{N}\sum_{i=1}^{N} c(v_i^j) \quad \text{s.t.} \quad c(v_i^j) = \Big\| f(v_i^j) - f(e^j) \Big\|_2 \tag{24}$$

The QuickDraw-FS dataset is built such that the category exemplar is located as closely as possible to the center of the category cluster. We plot in Figure P.1a, the average distance to the center as a function of the average distance to exemplar for each class and for **human**, the **CFGDM**, the **FSDM** and the **DDPM**. We observe a linear relationship between both distances (the lowest R^2 is 0.86). We also observe that the slope of the linear regression is not equal to 1 (≈ 0.67 for **human**, ≈ 0.61 for **CFGDM**, ≈ 0.58 for **DDPM** and ≈ 0.43 for **FSDM**). It suggests that there is not an exact match between the center of the category and the exemplar. We observe similar behavior in Figure P.1b, in which the distance value is averaged over the originality bins.



(a) averaged over object category

(b) averaged over originality bins

Figure P.1. Scatter plot of the distances to the center as a function of the distances to exemplar for the **humans**, the **CFGDM**, the **FSDM** and the **DDPM**. **(a)**: is averaged over object category, and **(b)**: is averaged over originality bins.

Q. Interpolation in the *generalization curves*

To obtain the data points of the generalization curves we compute the average originality and recognizability for all originality bins (see Section 4.2 for more information on the originality bins). We then interpolate between the data points using polynomial regression (degree 2). The fit of the polynomial curve is made using a least square error method. We report in Table 9.

Table 9. regression errors of the generalization curves

Network	γ	Least Square Error
human	-	4.1×10^{-7}
FSDM	-	9.4×10^{-5}
DDPM	0	3.7×10^{-5}
CFGDM	1	3.4×10^{-6}
CFGDM	0.2	3.7×10^{-5}
CFGDM	0.4	2.2×10^{-5}
CFGDM	0.6	1.3×10^{-5}
CFGDM	0.8	7.3×10^{-6}
CFGDM	1.5	1.3×10^{-6}
CFGDM	2.0	1.0×10^{-6}

R. More CFGDM importance feature maps

Using the Equation (4), we have computed the features importance map for 100 classes (see Figure R.1). For each class, we have averaged the importance maps obtained for 10 different samples generated by the CFGDM.

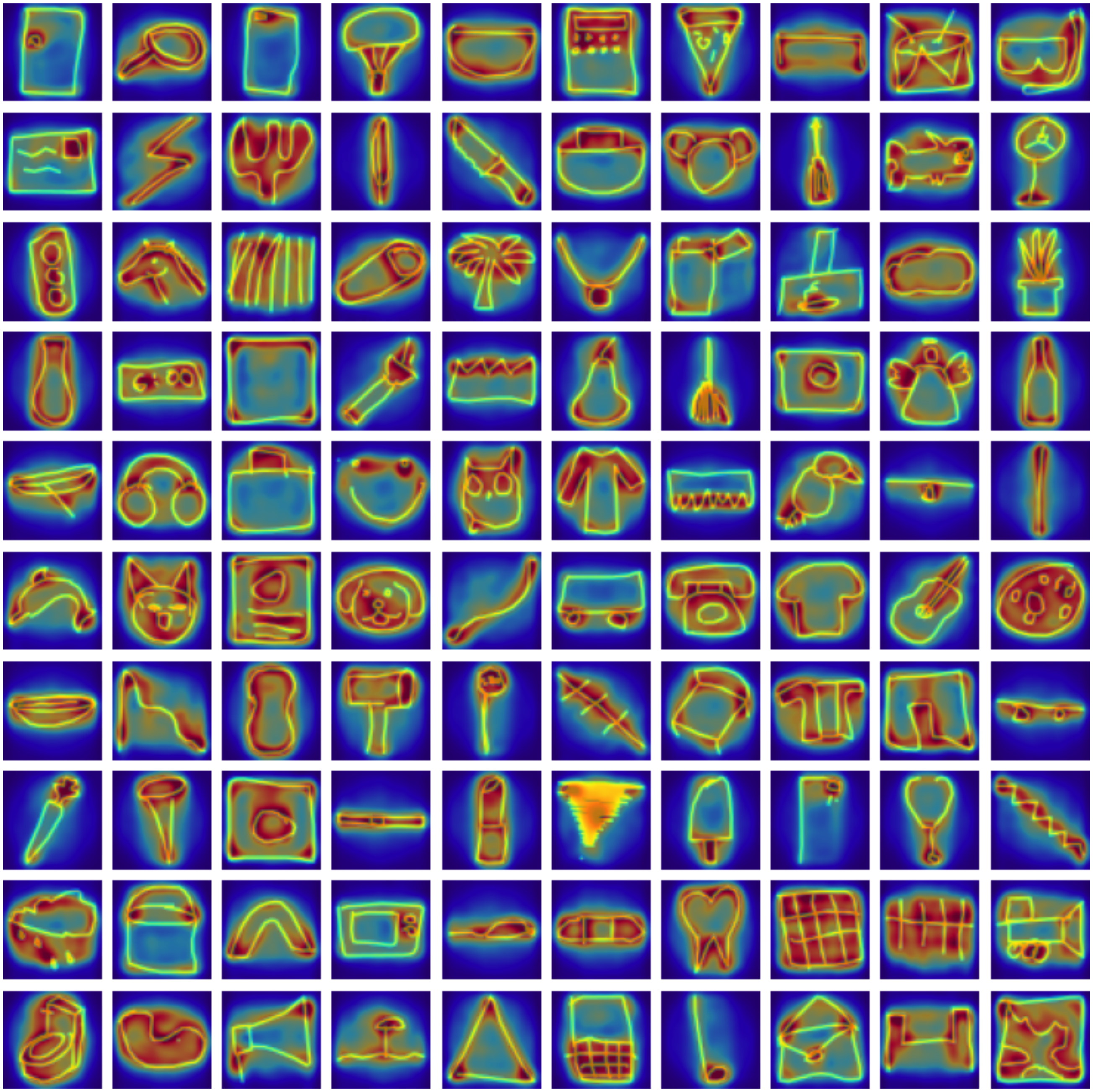


Figure R.1. CFGDM Importance feature map, for 100 categories. The maps are obtained by averaging n=10 misalignment maps $\phi(\mathbf{x}, \mathbf{y})$ as defined in Equation (4)

S. The ClickMe-QuickDraw Experimental Setup

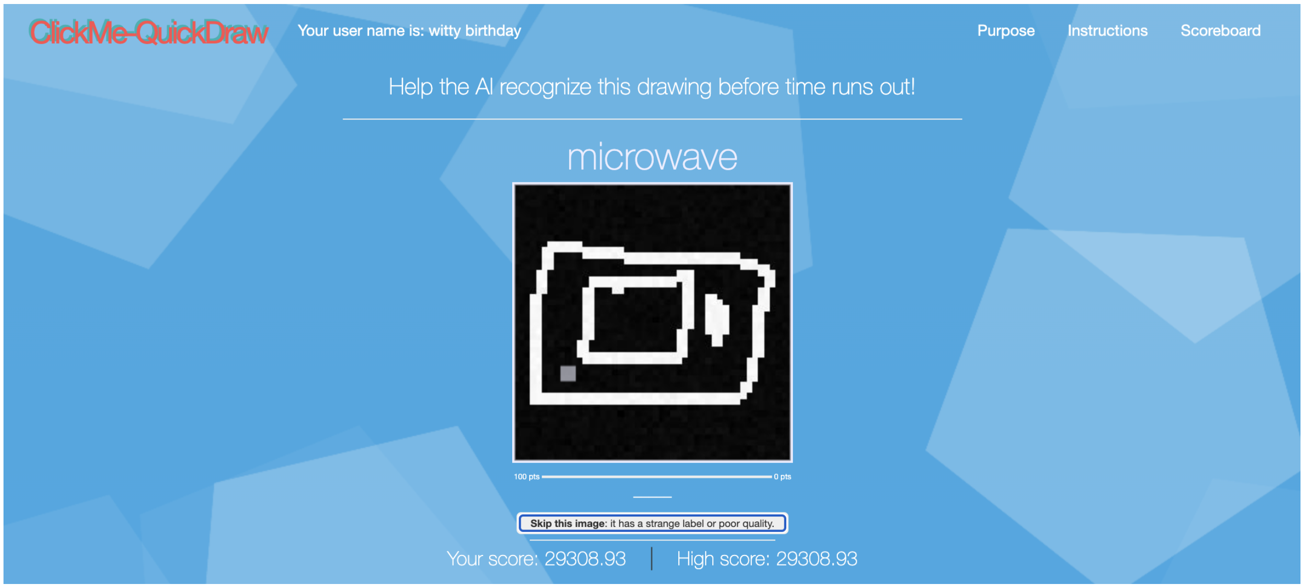

ClickMe-QuickDraw
Your user name is: witty birthday
Purpose
Instructions
Scoreboard
Help the AI recognize this drawing before time runs out!
microwave
100 pts
0 pts
Skip this image: it has a strange label or poor quality.
Your score: 29308.93
High score: 29308.93

Figure S.1. Screenshot of the ClickMe-QuickDraw web application

ClickMe-QuickDraw is a web application on which the user (alongside an AI model) plays to win prizes and help us understand differences in how humans and machines perceive drawings. The goal of the game is to help the AI partner recognize as many object drawings with as much confidence as possible.

The user helps the AI model recognize objects by revealing parts of images to it. This is done by *painting* over parts of the object image that help humans recognize it. When the user clicks on the image, the *brush stroke* begins dragging the cursor over the other important parts of the image. These image parts will be revealed to the AI model as the user paints over them, which will try to classify the drawing based on the regions that were *painted over*.

For every image guessed correctly, the user receives a score based on the time taken for the AI to recognize the image. The user will receive no points if the timer elapses before the AI model classifies correctly. The user can skip images that look strange or do not match their label by clicking the **Skip this image** button.

S.1. Web Application Design Parameters

Our ClickMe-QuickDraw experiment design followed the paradigm of ClickMe Linsley et al., 2018; images of object drawings are presented to the participant for 7 seconds before moving to the next image. We recruited 102 participants for this experiment, each instructed to annotate the most important parts of images of object drawings to complete the experiment. All the participants were informed of their participation in the experiment and the elementary drawing skill required. Participants were mostly selected among undergraduated students. The web app used for the experiment is built using Node.js and the production version of Python's Flask web server. As seen in Figure S.1, the user is presented with the correct class label and a 256×256 image of the drawing. The timer starts when the user clicks on the image and begins painting over the important parts of the image. As the user highlights parts of the image, the size of each click on the image is 21×21. The timer lasts for 5 seconds, and if the user fails to highlight parts that help the AI correctly classify the drawing, the drawing is skipped, and another one is displayed.

S.2. Classification Model

The AI model that classifies the highlighted parts of each image is a Lipschitz-constrained network trained to classify drawing images. The robustness of the model was imperative since we are working with model-generated images from a single prototype. The Lipschitz-constrained network yielded a classification accuracy of 0.99 on the entire database of images.

S.3. Data

The dataset for the web application contained 250 images, with 25 different classes and 10 images per class. The experiment had 102 participants, with 1050 correct annotations, giving us an average of 41.2 annotations across all 25 classes.

S.4. Reliability Analysis

We verify that the collected annotations (i.e., ClickMe maps) show a strong regularity and consistency across participants. We calculated the rank-order Spearman correlation between annotations from two randomly selected participants for an image. The annotations are blurred with a Gaussian kernel (of size 49×49). Such a blurring facilitates the comparison and reduces the noise related to the online application since the brush strokes are drawn with a computer mouse. We repeat this procedure $1\,0000$ times and average the per-image correlation. In Figure S.2, we plot the distribution of the per-image rank-order Spearman correlation.

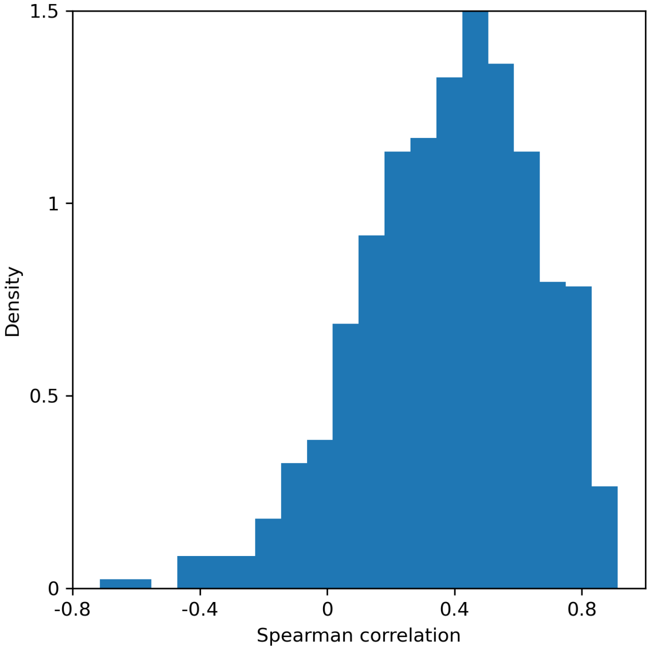


Figure S.2. Distribution of the per-image rank-order Spearman correlation.

We have filtered out the samples with a Spearman correlation 2 standard deviations away from the mean, which corresponds to a p-value of 5%. Such a filtering process allows us to remove inconsistent annotation maps. After removing the outliers, we have a mean per-image rank-order Spearman correlation of 0.47 ($p < 5e - 2$). This is to be compared to the mean Spearman correlation between 2 randomly selected annotations, which is 0.05.