OVERCOMING CHALLENGES IN LEVERAGING GANS FOR FEW-SHOT DATA AUGMENTATION

Christopher Beckham[1,2,3], **Issam Laradji**[1], **Pau Rodriguez**[1], **David Vazquez**[1], **Derek Nowrouzezahrai**[2,4], **Christopher Pal**[1,2,3,†]
[1]ServiceNow Research, [2]Mila, [3]Polytechnique Montreal, [4]McGill University, [†]Canada CIFAR AI Chair

ABSTRACT

In this paper, we explore the use of GAN-based few-shot data augmentation as a method to improve few-shot classification performance. We perform an exploration into how a GAN can be fine-tuned for such a task (one of which is in a *class-incremental* manner), as well as a rigorous empirical investigation into how well these models can perform to improve few-shot classification. We identify issues related to the difficulty of training and applying such generative models under a purely supervised regime with very few examples, as well as issues regarding the evaluation protocols of existing works. We also find that in this regime, classification accuracy is highly sensitive to how the classes of the dataset are randomly split. To address difficulties in applying these generative models under the few-shot regime, we propose a simple and pragmatic semi-supervised fine-tuning approach, and demonstrate gains in FID and precision-recall metrics as well as classification performance.

1 INTRODUCTION

In the past decade, deep learning has demonstrated immense success in many tasks and data modalities (Krizhevsky et al., 2012; LeCun et al., 2015; He et al., 2016). However, the issue of deep neural networks being highly data inefficient remains a major problem (Krizhevsky et al., 2012; Srivastava et al., 2014; Zhang et al., 2017; Tan et al., 2018), accentuated by the fact that for some domains, obtaining sufficient labelled data is laborious and expensive. This limits real-world applicability of deep neural networks, so one active area of research is in figuring out how to make these models transfer or learn better on under-represented datasets. Few-shot learning is one of the subfields that addresses this, where the goal is to facilitate adaptation to novel tasks with very few examples.

Few-shot learning as a field is rather broad (Wang et al., 2020), and many techniques can be seen as incorporating prior knowledge into either the data, the model/architecture, or the algorithm in order to make the learning process more sample efficient. For example, this may be in the form of multitask learning (Caruana, 1997; Zhang & Yang, 2017) where the learning of a few-shot task can be augmented by learning other relevant tasks with more data, or in embedding learning (Bertinetto et al., 2016; Sung et al., 2018; Vinyals et al., 2016) where a low-dimensional metric space is learned to be able to easily facilitate comparisons between unseen classes. In terms of algorithmic few-shot learning, meta-learning (Finn et al., 2017; Ravi & Larochelle, 2016), i.e. *learning how to learn*, can be used to learn more sample efficient optimisers.

In this paper, we focus on the *data* aspect of few-shot learning, which at its core leverages data augmentation to expand a small set of examples into a much larger set that will be sufficient to train deep neural networks. In particular, we explore data augmentation with *generative adversarial networks* (GANs) (Goodfellow et al., 2014), where the goal is to generate samples that are indistinguishable from the ones comprising the few-shot task of interest. Concretely, our contributions are as follows:

- We consider the entire few-shot data augmentation pipeline, where the goal is to pre-train a GAN on some set of source (training) classes, fine-tune it on very few examples of target (test) classes, and have it be able to generate new images from those classes to improve generalisation performance of a pre-trained classifier on the same target classes. Unlike previous works (Antoniou et al., 2017; Hong et al., 2020a;b), we specifically fine-tune our GAN on novel classes rather than try to generate from them in 'zero shot' fashion.
- We rigorously evaluate few-shot data augmentation performance by considering different randomised splits of the dataset (*dataset seeds*), where the splits are performed with respect to the *classes*. We also perform these experiments over different fine-tuning strategies where different parts of both the generator and discriminator are frozen, with the adversarial game still being played between both networks.

- We highlight the difficulties involved in few-shot data augmentation, and explain limitations with the empirical evaluation of existing work. To this end, we propose a more pragmatic strategy involving semi-supervised GAN fine-tuning, and demonstrate performance gains with respect to classification performance, as well as FID and precision/recall. As a side contribution, one of our proposed fine-tuning strategies enables the generator to learn new classes in a *class-incremental* manner, without catastrophic forgetting of old ones. This may be of interest to those working in the intersection of generative modelling and *generalised* few-shot learning.

1.1 Few-shot data augmentation pipeline

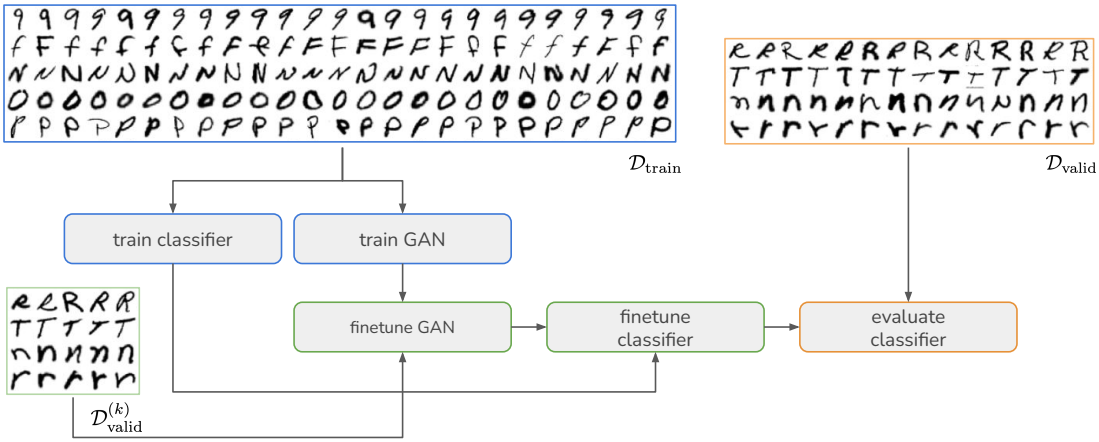


Figure 1: The few-shot data augmentation pipeline. In this illustration, the training set consists of five classes (abundantly represented) and the validation set consists of four classes. The support set is a subset of the validation set and only comprises five examples per class ($k = 5$). Each stage is described in Section 2.

Our few-shot data augmentation pipeline is illustrated in Figure 1. First, we divide the data into a training set $\mathcal{D}_{\text{train}}$, validation set $\mathcal{D}_{\text{valid}}$, and test set $\mathcal{D}_{\text{test}}$, where $\mathcal{Y}_{\text{train}} \cap \mathcal{Y}_{\text{valid}} = \emptyset$ and $\mathcal{Y}_{\text{valid}} = \mathcal{Y}_{\text{test}}$. We can think of the training and validation (test) sets as comprising our *source* and *target* classes, respectively. We also assume that each source class is abundantly represented. Our goal is to train a conditional generative model that can generate plausible examples from the validation set $\mathcal{D}_{\text{valid}}$ given only a very small subset of that same set, which is called the 'support set' $\mathcal{D}^{(k)}_{\text{valid}}$. The superscript (k) denotes that there are only k examples per target class that we can work with. Because this is an extremely difficult task, we pre-train the GAN on the source classes first (i.e. the training set), which we assume is abundantly represented. Our primary measure of success is whether the examples generated from our model are able to improve the generalisation performance of a classifier trained to predict the validation classes $\mathcal{Y}_{\text{valid}}$.

Because the few-shot setting requires us to adapt our models to novel classes with very small amounts of data, the evaluation can be highly sensitive to how the dataset is split. Furthermore, it may also be sensitive to what classes comprise source classes (those in the training set) and what classes comprise novel classes (those in the validation or test set). Because of this, almost all quantitative results in this paper are averages over many randomised splits of the dataset in order to produce reliable estimates of uncertainty. (This is illustrated in Figure S18.)

2 Proposed method

We break down our training and evaluation pipeline into the following subsections:

1. Section 2.1: pre-train a classifier on $\mathcal{D}_{\text{train}}$ which estimates $p(\boldsymbol{y} \in \mathcal{Y}_{\text{train}}|\boldsymbol{x})$. This classifier will eventually be fine-tuned to adapt to the new classes in $\mathcal{D}_{\text{valid}}$.
2. Sections 2.2 and 2.3: train a conditional GAN on $\mathcal{D}_{\text{train}}$, where we learn a conditional generative model to be able to sample $\boldsymbol{x} \sim p_G(\boldsymbol{x}|\boldsymbol{y} \in \mathcal{Y}_{\text{train}})$. Afterwards, we leverage some *fine-tuning strategy* to fine-tune the GAN on $\mathcal{D}^{(k)}_{\text{valid}}$, which will allow sample generation for the novel classes in $\mathcal{Y}_{\text{valid}}$.

3. Use the fine-tuned GAN to generate new examples for each *target class*. These examples will be concatenated with the original support set $\mathcal{D}_{\text{valid}}^{(k)}$ to produce an 'augmented' set which we denote $\tilde{\mathcal{D}}_{\text{aug}}^{(k)}$.

4. Section 4: Replace the pre-trained classifier's output probability distribution to be over the novel classes $p(\boldsymbol{y} \in \mathcal{Y}_{\text{valid}}|\boldsymbol{x})$ and fine-tune it by using $\tilde{\mathcal{D}}_{\text{aug}}^{(k)}$ as the training set and perform model selection (hyperparameter tuning) on $\mathcal{D}_{\text{valid}}$. The best model is evaluated on $\mathcal{D}_{\text{test}}$, which is the unbiased measure of generalisation performance.

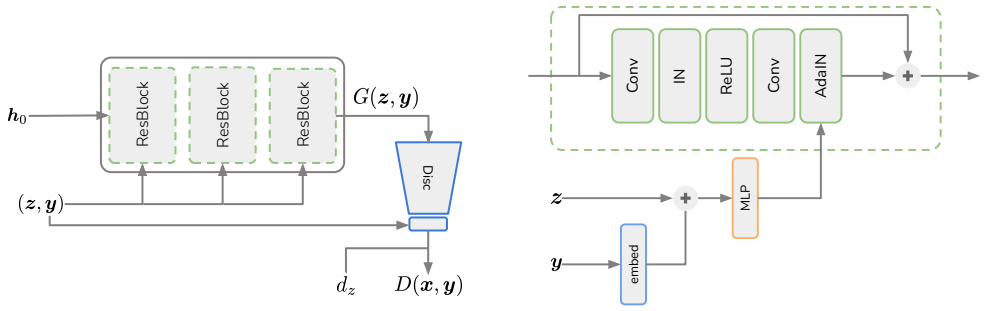
2.1 Classifier pre-training

Before we train our generative model, we need to train a classifier which can easily be adapted to any new classes we encounter. To do this, we first need to train it on a large dataset where there exists many examples per class, so that the features learned are robust and reliable. In our case, this is the training set $\mathcal{D}_{\text{train}}$.

The classifier we train is a ResNet-50 (He et al., 2016) initialised from scratch, trained with ADAM (Kingma & Ba, 2014) using learning rate 1×10^{-4} and moving average coefficients $\beta = (0.9, 0.999)$. The training set is split into a 95%-5% split, with the latter forming an internal validation set for early stopping. We train this classifier with a moderate amount of data augmentation, which can comprise randomly-resized crops anywhere between 70% to 100% of the original image size, and random rotations anywhere between -10 and 10 degrees. We simply train the classifier using the small internal validation set as an early stopping criteria.

2.2 Generative adversarial networks

The generative adversarial network (Goodfellow et al., 2014) we use here is based on the projection discriminator proposed by Miyato & Koyama (2018), with a slight modification to make it more similar to the recently proposed StyleGAN class of models (Karras et al., 2018). In StyleGAN, rather than the latent code being fed directly as input to the generator network, the input is instead a learnable constant tensor $\boldsymbol{h}_0$ whose progressively growing representation is modulated by the latent code $\boldsymbol{z}$ via adaptive normalisation layers. This decision was made to more easily facilitate fine-tuning to new classes, since the adaptive normalisation layers comprise relatively few parameters compared to the main backbone. However, it has also been shown in Karras et al. (2018) that this particular architecture is superior to traditional architectures in terms of image quality and disentanglement.



(a) (b)

Figure 2: **Left:** overview of both the generator and discriminator. **Right:** The residual block used for the generator. The label $\boldsymbol{y}$'s embedding is extracted and concatenated with the latent code to produce scale and shift coefficients for the second (last) instance normalisation layer. The addition symbol for the $(\boldsymbol{z}, \boldsymbol{y})$ part of the architecture indicates tensor concatenation.

We train a conditional GAN which comprises two networks: the generator $G(\boldsymbol{z}, \boldsymbol{y})$ and discriminator $D(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{z} \sim p(\boldsymbol{z}) = \mathcal{N}(0, \boldsymbol{I})$ is the latent code and $\boldsymbol{y}$ is the label. In terms of label conditioning, we utilise embedding layers that map indices from the label distribution $p(\boldsymbol{y})$ to codes which are concatenated with the latent codes $\boldsymbol{z}$, which in turn are used to modulate residual blocks in the generator via adaptive instance normalisation. This is illustrated in Figure 2. The objective function we use is the non-saturating logistic loss originally proposed in Goodfellow et al.

(2014):

$$\min_D \mathcal{L}_D = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p(\boldsymbol{x},\boldsymbol{y})} \log\left[D(\boldsymbol{x},\boldsymbol{y})\right] - \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{y}\sim p(\boldsymbol{y})} \log\left[1 - D(G(\boldsymbol{z},\boldsymbol{y}),\boldsymbol{y})\right] \tag{1}$$

$$\min_G \mathcal{L}_G = -\mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}),\boldsymbol{y}\sim p(\boldsymbol{y})} \log\left[D(G(\boldsymbol{z},\boldsymbol{y}),\boldsymbol{y})\right], \tag{2}$$

where $D = \text{sigm}(d(\boldsymbol{x},\boldsymbol{y}))$, with $d()$ denoting the raw logits of the output. In practice, we experienced an issue where the GAN would exhibit serious mode dropping. This appears to be because of our use of a learnable constant as input to the network, though it is currently unclear what architectural choices in StyleGAN mitigate this. To fix this, we utilise the InfoGAN (Chen et al., 2016) loss where the discriminator has to also predict the latent code $\boldsymbol{z}$ from the generated image $G(\boldsymbol{z},\boldsymbol{y})$, which can be used to maximise mutual information between the generated image and $\boldsymbol{z}$. This extra loss is $\|d(G(\boldsymbol{z},\boldsymbol{y}))_z - \boldsymbol{z}\|^2$ and is minimised wrt both networks and is weighted by coefficient γ, where $d(\cdot)_z$ is a $\boldsymbol{z}$ prediction branch that stems off the main backbone of d.[1]

2.3 Fine-tuning

When GAN training has completed, we need to figure out how to best fine-tune it on our support set, which contains very few examples (only k) per class. Here, our fine-tuning still involves optimising the adversarial losses in Equation 1 as opposed to only fine-tuning the generator with non-adversarial losses (such as in Li et al. (2019)). However, if we do not restrict the number of parameters involved during optimisation this can easily lead to overfiting. Since we are using the projection discriminator of Miyato & Koyama (2018), the discriminator's output logits can be written as $d(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{y}^T\boldsymbol{V}\cdot\phi(\boldsymbol{h}) + \psi(\phi(\boldsymbol{h}))$, where $\boldsymbol{h} = f_D(\boldsymbol{x})$ is the output of the discriminator's backbone and $\boldsymbol{V}$ is the embedding matrix that maps a (one-hot encoded) label to its embedding. From this equation, we can devise three ways in which D could be finetuned, and we call this hyperparameter *dfm* ('D finetuning mode'): *dfm=embed* corresponds to only updating $\{\boldsymbol{V}, d(\cdot)_z\}$; *dfm=linear* corresponds to updating $\{\boldsymbol{V}, \phi, \psi, d(\cdot)_z\}$; and *dfm=all* corresponds to updating the backbone f_D as well. We can also define the 'G finetuning mode' (*gfm*) in a similar manner, with *gfm=embed* denoting that we only update the embedding matrix per residual block, and *gfm=linear* denoting both the embedding matrix and the MLP which produces adaptive instance norm parameters (see Figure 2b). These choices will be compared against each other in Section 4.

3 Related work

A very closely related work is DAGAN (Antoniou et al., 2017). The authors proposed training a GAN where the discriminator is trained to distinguish pairs of images, rather than images conditioned on labels.. This means that D is trained to distinguish between a same-class tuples from the real distribution $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and a pair from the fake distribution, where one of the elements is generated $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$, where $\tilde{\boldsymbol{x}} = G(r(\boldsymbol{x}), \boldsymbol{z}))$. In this case, $r(\boldsymbol{x})$ is an encoder network and G is the corresponding decoder. The authors motivate this by wanting a discriminator that can naturally generalise to new classes, since it is not explicitly conditioned on a label. For their generator network, $r(\boldsymbol{x})$ is meant to infer (without any explicit label supervision) the 'content' of the image, while $\boldsymbol{z}$ serves as the 'style' which is simply noise injected from a prior. However, upon attempting to reproduce this model we found a major deficiency, in that the trained decoder G ignores $r(\boldsymbol{x})$ completely, i.e $G(\boldsymbol{r}, \boldsymbol{z}) \approx G(\boldsymbol{0}, \boldsymbol{z})$. This seems most likely attributable to the fact there is no supervision at all for $r(\boldsymbol{x})$, i.e. a reconstruction error or InfoGAN-style losses to ensure that both $\boldsymbol{z}$ and $r(\boldsymbol{x})$ are used by the network. Such losses do not appear to be present the paper. Furthermore, from the results it is unclear if the method would outperform a discriminator that conditions on the class label for the training set. Despite this, a big difference between their works and ours is that they try to perform zero-shot generation with respect to the *generative model*, since it itself is not fine-tuned on the novel classes. In our own preliminary experiments however a model we trained similar to DAGAN, we found that when we tried to generate images from novel classes with $r(\boldsymbol{x})$, there was a strong bias towards it generating images that looked like classes from the training set.

F2GAN (Hong et al., 2020b) proposes an adversarially-augmented autoencoder where k-way mixup (Zhang et al., 2017; Verma et al., 2019a) is performed between the latent features of k images, coupled with an attention module to patch up discrepencies in the image (hence 'fusing and filling'). Unlike DAGAN, a conditional discriminator is used which conditions on labels. In F2GAN, a 'mode seeking' loss is proposed to ensure that, for the same pairs of images, different mixing coefficients correspond to sufficiently diverse images. While an ablation study was performed to justify each component/loss that was added, this was not done for any of the datasets for which few-shot classification was performed and therefore it is unclear which components of the model are most influential. Like DAGAN, there is

[1] We suspect this loss may have a similar effect to the minibatch standard deviation layer in StyleGAN, which allows the discriminator to distinguish between real/fake based on minibatch statistics.

no finetuning on the support set of novel classes, and the model is expected to perform zero-shot generation. We do not pursue zero-shot generation in this paper because we found it is extremely difficult to expect GANs to generalise to unseen classes, a concern that is also shared by Wertheimer et al. (2020). It is likely that the multitude of losses proposed in F2GAN are there to inject strong priors into the network such that it can perform well for zero-shot generation, but in this work we take a more direct approach where we fine-tune our GAN on few-shot classes. (Additional discussion regarding zero-shot generation for GANs (and VAEs) can be found in Section S8.1.)

One major issue in comparing these papers' results is that there is no standardised way in which the data is split for training and evaluation. DAGAN is the most rigorous in its evaluation, utilising a 'source' domain (training set), a 'validation' domain (validation set), and a 'target' domain (test set). For the validation and testing sets, k examples per class are held out in each, which can be thought of as the support sets for both the validation and testing sets, respectively. These support sets are leveraged by the generative model to create more examples for their respective classes. In DAGAN, the validation set is used to tune the number of synthetic examples that should be generated per class. Conversely, in F2GAN there appears to be no mention of a validation set, and the data is simply split between a training domain and a testing domain. Beause of hyperparameter tuning, it is likely performance estimates are biased. This will be discussed in further detail in Section 4.

There are many works which involve the fine-tuning of GANs. Robb et al. (2020) proposes a new method to finetune a pre-trained StyleGAN (Karras et al., 2018) in the few-shot regime by tuning only the singular values of both networks. Karras et al. (2020) propose a trick to mitigate discriminator overfitting to improve StyleGAN2 training, and present transfer learning results in the context of domain adaptation. Such tricks could be leveraged to potentially improve our results in Section 2.3. The aforementioned work of Wertheimer et al. (2020) also performs few-shot data augmentation but in the context of domain adaptation, where one is interested in mapping between datasets. Casanova et al. (2021) propose 'instance conditioned' GANs that condition on pre-trained self-supervised embeddings rather than class labels, and demonstrate impressive zero-shot transfer to other datasets, simply by computing embeddings over these other datasets. While these works undoubtedly present impressive results on larger and more ambitious datasets, none of them have examined a few-shot pipeline where the goal is to generate images to improve a downstream task directly, such as classification performance. These works instead primarily report image similarity metrics computed on their generated images.

A closely related subfield is that of few-shot hallucination (Hariharan & Girshick, 2017), which may also involve using generative models (Li et al., 2020) to 'hallucinate' (generate) additional examples for novel classes, but in latent space instead of input space. Similar to our procedure, these hallucinated latent samples are combined with real latent samples to make one larger dataset. In principle, our method could be adapted to generate latent codes (with respect to a latent distribution), but this is left to future work. Lastly, while we do not consider meta-learning here, our few-shot pipeline could be framed as an end-to-end approach under that paradigm. There are numerous works combining generation with few-shot classification (Zhang et al., 2018; Wang et al., 2018; Chen et al., 2019; Verma et al., 2019b).

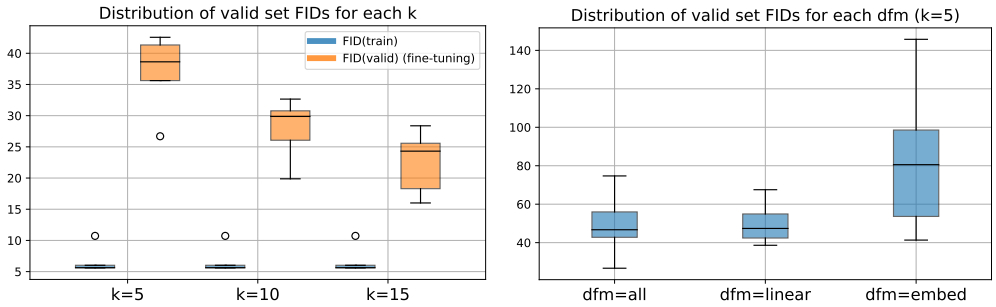
4 Experiments and Results

We run experiments over five randomised dataset seeds (splits) on the balanced version of the EMNIST dataset (Cohen et al., 2017). For each dataset seed, the training set and validation sets comprise of 38 and 9 randomly sampled classes (without replacement), respectively. Each class comprises 2800 examples, with 80% being allocated for validation and 20% for testing. The support set is a subset of the validation set, with only k examples per class. The test set shares the same classes as the validation set but is held out and is only used for unbiased evaluation for classifier fine-tuning in Section 4.1. Next, we summarise our findings with regards to the stages proposed in Section 2.

GAN pretraining (Section 2.2): We found that our InfoGAN coefficient γ does not have to be anywhere close to zero for it to have its intended effect of mitigating mode dropping. We train with ADAM (Kingma & Ba, 2014) parameters $\beta = (0.0, 0.9), \text{lr} = 2 \times 10^{-4}$, InfoGAN coefficient $\gamma = 100$, with a G:D update ratio of 1:5. For early stopping, we monitor the FID (Heusel et al., 2017) between our generated samples and training set. More specifically, we randomly sample $N = 5000$ examples from the training set to compute the statistics of the reference distribution, and with the labels extracted from those examples we conditionally generate samples from our GAN which comprises the generative distribution. This metric serves as our early stopping criteria.

GAN finetuning (Section 2.3): For our fine-tuning strategy, we found that the best strategy is to optimise *only the embedding layers* in G and, quite surprisingly, to optimise for *all* parameters in D, using the same adversarial losses and the same InfoGAN coefficient γ. In order to facilitate new classes, all that is required is for new indices (rows) to

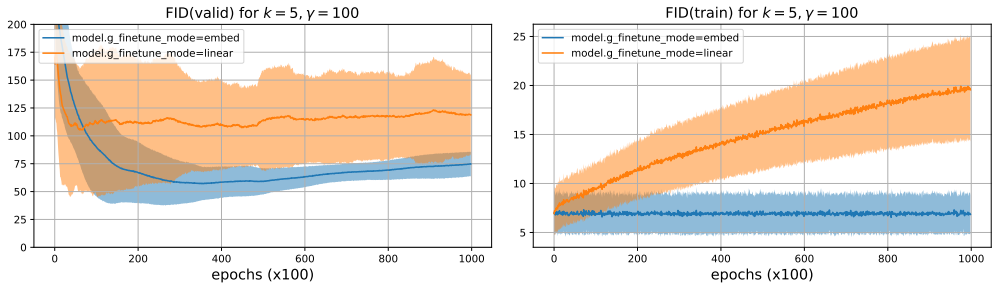
be added to each and every embedding layer's matrix in both networks.[2] Here, our early stopping criteria is the FID on the *validation set*; that is, we randomly sample N examples from the validation set as our reference set and use the labels from those examples for the generative distribution. In Figure 3a we show the distribution of valid set FIDs as a function of k. Perhaps not surprisingly, as k gets larger we are able to generate samples that progressively become more representative of those in the validation set. Here, we also show the FID on the training set (before fine-tuning) as a reference.



(a) (b)

Figure 3: **Left**: For each k, we show the distribution of FID valid set scores (over dataset seeds) when the GAN is fine-tuned on the support set $\mathcal{D}^{(k)}_{\text{valid}}$. For each of these k, the training set FID is shown as blue as a reference. **Right**: the distribution of FID valid set scores (over dataset seeds and *gfm* modes) for different *dfm* modes.

For the sake of comparison, we also measure performance across different *dfm*'s. In Figure 3b we plot FID on the validation set for each of the three *dfm* modes, with the distribution of FIDs being computed over both dataset seeds and *gfm* . We can see that *dfm=embed* performs the worst, and surprisingly, not much difference between *dfm=all* and *dfm=linear*. We find this to be an interesting result given it is the opposite of 'Freeze-D' (Mo et al., 2020), whose work found the best results in fine-tuning *all* of G and just the later layers of D.



(a) *gfm* $\in$ {linear, embed}, valid FID (b) *gfm* $\in$ {linear, embed}, train FID

Figure 4: **Left**: FID between generated samples on the target classes and the validation set, for GAN fine-tuning (lower FID is better). The orange curve corresponds to our proposed fine-tuning mode for G (embedding layers only, i.e. *gfm=embed*) versus also fine-tuning the MLPs which produce adaptive instance norm coefficients (i.e. *gfm=linear*, see Figure 2b). Shaded regions (variance) is over all possible *dfm*'s (*all*, *linear*, or *embed*). **Right**: FID but computed between samples on the source classes and the training set. Here, we can see that fine-tuning only the embedding layers in G does not result in catastrophic forgetting in how to generate samples for source classes.

In Figure 4 we present some additional results comparing the fine-tuning mode of G, which we denote as *gfm*. We compare fine-tuning just the embeddings in G (*gfm=embed*) versus also fine-tuning the MLPs that take part in adaptive instance normalisation (*gfm=linear*, see Figure 2b). It can seen that with *gfm=embed*, validation set FIDs are superior on average and that the training set FID does not degenerate over time (Figure 4b). One may wonder in what situations does it simply suffice to only finetune the embedding layers in G and have it perform well for generation. We recognise that this is highly dependent on both the architecture and datasets used. Unlike the more ambitious task of domain

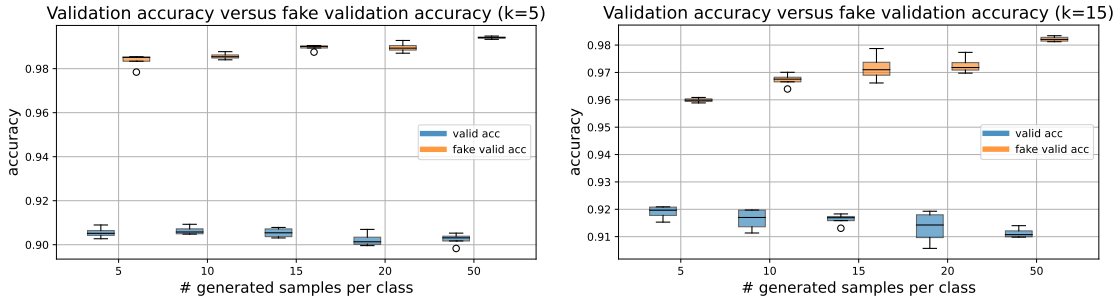
[2]Implementation-wise, this is not neccessary as we simply assume that we know the total number of classes beforehand (train+valid+test) and initialise the embedding matrix to have precisely this many rows.

transfer, here our source and target classes can be seen as coming from the same overall distribution $p(\boldsymbol{x})$. If source classes in the training set share very similar factors of variation to target classes in the validation or test set, then we would expect the network to generalise to those new classes with minimal modification to the generator network.

4.1 Classifier finetuning

Here, we present the results of our classifier fine-tuning experiments on both the validation and test sets. While we already have a fine-tuned GAN which can generate images from the novel classes, we also need to fine-tune the classifier which was originally trained on the source classes (Section 2.1). To do this, we simply freeze all of its parameters and replacing its output (logits) layer with a new layer which defines a probability distribution over either the validation classes $\mathcal{Y}_{\text{valid}}$ or the test classes $\mathcal{Y}_{\text{test}}$. Note that unlike with generalised few-shot learning, here we are not interested in maximising performance over both the old and new classes – we simply wish to maximise performance over the latter. Here, we use the same parameters for ADAM as described in Section 2.1 but lower the learning rate to 2×10^{-5}, and train for 30k epochs. We describe our experiments and their corresponding hyperparameters as follows:

- **Baseline**: we fine-tune the pre-trained classifier on $\mathcal{D}^{(k)}_{\text{valid}}$. We run two experiments for this, one with and without traditional data augmentation applied. For the data augmentation experiment, we tune over the minimum random resized crop size `minScale` $\in \{0.2, 0.4, 0.6, 0.8\}$.
- **Mixup**: use mixup in input space (Zhang et al., 2017), where the mixing distribution Beta(λ, λ) is a hyperparameter, and $\lambda \in \{0.1, 0.2, 0.5, 1.0\}$. This is done 'on the fly': for each minibatch seen during fine-tuning, input mixup is applied to both the images and their labels. `minScale` is fixed to 0.8 here.
- **Fine-tuned GAN**: Using the fine-tuned GAN, we pre-generate n_s new samples per class and combine these with the original supports $\mathcal{D}^{(k)}_{\text{valid}}$ to create $\tilde{\mathcal{D}}^{(k)}_{\text{aug}}$. Here, $n_s \in \{2, 5, 10, 20\}$ is a hyperparameter that we explore. `minScale` is also fixed to 0.8 here.



(a) $k = 5$ (b) $k = 15$

Figure 5: Boxplot showing validation accuracy (blue) and fake validation accuracy (orange) for different numbers of generated samples per class n_s. Fake validation accuracy is a held-out validation set generated by the GAN. It can be seen that as n_s increases, the actual validation accuracy decreases and fake validation accuracy increases. Results are shown over one dataset seed, with each boxplot being a distribution over the different sampling runs.

We present our results in in Table 1. We can see that all results are characterised by non-negligible variances, especially when k is small. This can be problematic because it can diminish the statistical significance of any claims in performance gains. If we just consider the mean accuracy however, the data augmentation baseline performs best for $k = 5$ and $k = 15$ and our method performs best for $k = 10$. We found that validation accuracy is highly dependent on the number of generated samples per class n_s, and often times accuracy degrades when it is too high. We conjecture that this is because there is a fundamental mismatch between the distribution of GAN-generated images and those from the validation set (whose accuracy we wish to maximise), and this is leading to overfitting on the former. To validate this, we generate a held-out validation set from the same GAN that we generated the images from, which we call our 'fake' validation set. We can see in Figure 5 that as n_s increases, so does accuracy on our fake validation set, and this is negatively correlated with accuracy on the actual validation set. This clearly indicates we are indeed overfitting to the distribution induced by our GAN.

The results reported by F2GAN (Hong et al., 2020b) in the upper panel of Table 1 are indeed notable compared to ours for $k = 10$ and $k = 15$, but unfortunately their description of how the dataset was split is unclear and there are concerns with how principled this evaluation is. In their work, they describe EMNIST as comprising of 47 classes

Table 1: Accuracy (%) of different methods on EMNIST on $\mathcal{D}_{\text{valid}}$. The separating bar distinguishes between results we have lifted from the F2GAN (Hong et al., 2020b) paper and our own results. These two groups of results are not directly comparable due to a variety of confounding factors in the experimental setup. In our results, uncertainty estimates are computed over five different randomised dataset splits (seeds) on the validation set. For each of these seeds, hyperparameter tuning was used to obtain the best result. The best hyperparameters can be found in Section S8.6.

	Method	EMNIST $k = 5$	EMNIST $k = 10$	EMNIST $k = 15$
F2GAN	Baseline	83.64	88.64	91.14
	w/ data aug.	84.62	89.63	92.07
	FIGR (Clouâtre & Demers, 2019)	85.91	90.08	92.18
	GMN (Bartunov & Vetrov, 2018)	84.12	91.21	92.09
	DAWSON (Liang et al., 2020)	83.63	90.72	91.83
	DAGAN (Antoniou et al., 2017)	87.45	94.18	95.58
	MatchingGAN (Hong et al., 2020a)	91.75	95.91	96.29
	F2GAN (Hong et al., 2020b)	**93.18**	**97.01**	**97.82**
ours	Baseline	90.97 ± 4.62	92.68 ± 4.01	93.45 ± 3.69
	w/ data aug.	**91.27 ± 4.72**	92.77 ± 4.06	**93.47 ± 3.80**
	Input mixup (Zhang et al., 2017)	91.00 ± 4.39	92.68 ± 3.99	93.32 ± 3.68
	Fine-tuned GAN	91.11 ± 4.62	**92.85 ± 4.02**	93.35 ± 3.79
	+ semi-supervised (Sec. 5)	**92.12 ± 4.34**	**93.27 ± 4.00**	**93.79 ± 3.72**

Table 2: Results of selected experiments on the held-out test set $\mathcal{D}_{\text{test}}$. For each experiment and its dataset seed, the hyperparameters used for the test set evaluation correspond to the optimal hyperparameters found for the same corresponding experiment / seed on the validation set (e.g. # of epochs, α for input mixup, n_s for GAN).

	Method	EMNIST $k = 5$	EMNIST $k = 10$	EMNIST $k = 15$
ours	Baseline w/ data aug	90.65 ± 4.52	92.22 ± 4.19	93.02 ± 3.94
	Input mixup (Zhang et al., 2017)	90.69 ± 4.34	92.27 ± 4.05	93.04 ± 3.65
	Fine-tuned GAN	90.81 ± 4.53	92.50 ± 4.11	92.90 ± 4.03
	+ semi-supervised (Sec. 5)	**91.64 ± 4.35**	**92.95 ± 4.14**	**93.42 ± 3.84**

but with 28 being selected as source (training) classes and 10 as targets (testing), leaving 9 classes unaccounted for. In their preceding work MatchingGAN (Hong et al., 2020a), the dataset is split into 28-10-10 (i.e. train-valid-test) for a total of 48 classes, with 10 classes being used as a validation set for GAN training. The test set (10 classes) is used for classifier evaluation, with a small support set of k examples per class used by the GAN to generate additional examples. However, excessive tuning of hyperparameters of the GAN (that is, the hyperparameters that directly control generation, e.g. the generation seed or number of generated samples per class n_s) and hyperparameters of the classifier can lead to biased estimates of performance on the test set, which is why we have also presented results on our test set in Table 2. In this table, we find that for all values of k, the data augmentation baseline performs the best in terms of mean accuracy, though like we have mentioned earlier, these results have relatively inflated variances which diminish their significance.

Another confounder is that MatchingGAN report artificially reducing the size of all classes in EMNIST to be 100 examples per class to mimic DAGAN's setup, but it is not clear whether F2GAN has also done this. For reference, the only two splits of EMNIST with 47 classes are `ByMerge` and `Balanced`, with `Balanced` containing 2,800 examples per class and `ByMerge` containing anywhere between 2,961-44,704 examples per class, so this is an enormous reduction in dataset size. Such a reduction would be detrimental to both classifier pre-training and GAN training. While DAGAN's evaluation protocol differs slightly to our work and the other works, they report 76% test set accuracy on EMNIST with $k = 15$ with this artificial reduction in examples per class. Lastly, for both MatchingGAN and F2GAN, the number of samples generated per class with the GAN was set to $n_s = 512$, which is extraordinarily large, considering that we obtain worse results on our validation set with anything more than $n_s = 5$, on average. Our results appears to corroborate that of DAGAN, whose authors report only tuning $n_s \in \{1, \ldots, 10\}$. Because there are many subtle differences in the empirical evaluation between our work and the aforementioned ones, we simply defer the reader to Figures S18, S19, S20 for more information rather than enumerate all of these here.

Lastly, we also present results on Omniglot (Lake et al., 2015), where each dataset seed comprises 1411 source classes and 212 target classes. Note however that Omniglot does not fit nicely with our particular training paradigm, which assumes that the source classes are abundant in their number of examples. In Omniglot, while there are many classes (1623), there are only 20 examples per class, which means that we are operating in a low data regime even when we are pre-training our GAN on the source classes. While we achieve the highest validation accuracy here, the unbiased (test set) evaluation favours the data augmentation baseline.

Table 3: Results for Omniglot, for $k = 5$. Results are averages over five different dataset seeds.

	Method	$\mathcal{D}_{\text{valid}}$	$\mathcal{D}_{\text{test}}$
ours	Baseline w/ data aug	97.35 ± 0.33	$\mathbf{95.56 \pm 0.47}$
	Input mixup	96.85 ± 0.34	94.95 ± 0.86
	Fine-tuned GAN	97.33 ± 0.32	95.49 ± 0.58
	+ semi-supervised (Sec. 5)	$\mathbf{97.40 \pm 0.29}$	95.28 ± 0.65

5 Further discussion

One major limitation of using conditional GANs for data augmentation is that there is no way to leverage unlabelled data, which tends to be more abundant than labelled data. Another issue with our setup is that downstream performance is likely to only give practical gains for a very specific range of values for k. For example, when k is too small, there is a strong incentive for one to leverage GAN-based data augmentation since there are very few examples per class, but as we have demonstrated, fine-tuning a GAN is difficult precisely because of how few examples there are. Conversely, as k becomes larger, even if we could finetune a better GAN we would also expect there to be diminishing returns when it comes to improving classification performance over the baseline, since there are an abundant number of labelled examples per class. Therefore, having the ability to leverage unlabelled data seems like a more pragmatic endeavour since we could still experiment with small k but likely do a much better job at fine-tuning a better quality generative model over the new classes. Fortunately, it turns out that one can take the projection discriminator equation of Miyato & Koyama (2018) that we use and easily turn it into a semi-supervised variant. Recall from Equation 3 that the output logits of the discriminator can be written as follows:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{y}^T \boldsymbol{V} \cdot \phi(\boldsymbol{h}) + \psi(\phi(\boldsymbol{h})), \tag{3}$$

with $\boldsymbol{h} = f_D(\boldsymbol{x})$, where f_D is the backbone (feature extractor) of the discriminator. From Miyato & Koyama (2018), this equation is the result of trying to model the likelihood ratio between the real and generative distributions, which in turn can be decomposed into two ratios that correspond to the terms in Equation 3:

$$d(\boldsymbol{x}, \boldsymbol{y}) \approx \log \frac{q(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x}, \boldsymbol{y})} = \log \frac{q(\boldsymbol{y}|\boldsymbol{x})q(\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x})} = \underbrace{\log \frac{q(\boldsymbol{y}|\boldsymbol{x})}{p(\boldsymbol{y}|\boldsymbol{x})}}_{\approx \boldsymbol{y}^T \boldsymbol{V} \cdot \phi(\boldsymbol{h})} + \underbrace{\log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}}_{\approx \psi(\phi(\boldsymbol{h}))}, \tag{4}$$

From this, one can easily see that the last term in Equation 4 models log ratio $q(\boldsymbol{x})/p(\boldsymbol{x})$ which is not dependent on any label. From this, we could play an additional adversarial game between G and D for just this term. If we overload notation so that $D(\boldsymbol{x}) = \text{sigm}(d(\boldsymbol{x})) = \text{sigm}(\psi(\phi(\boldsymbol{h})))$ we can derive a semi-supervised set of losses:

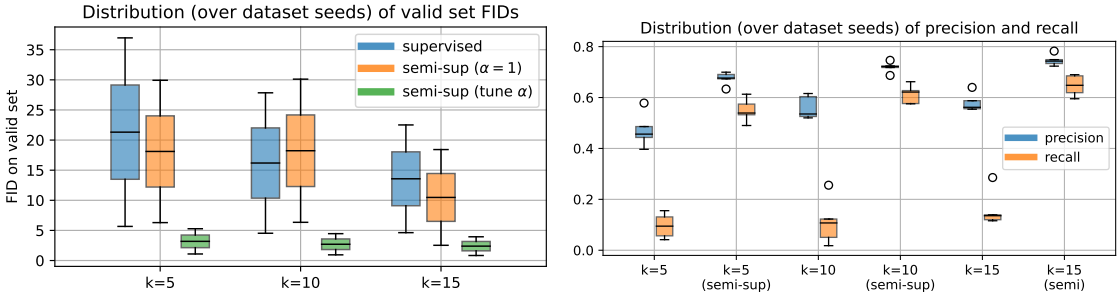
$$\min_D \mathcal{L}_D = \mathcal{L}_D^{(\text{sup})} - \alpha \Big[\mathbb{E}_{\boldsymbol{x} \sim p_u(\boldsymbol{x})} \log \big[D(\boldsymbol{x}) \big] - \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}), \boldsymbol{y} \sim p(\boldsymbol{y})} \log \big[1 - D(G(\boldsymbol{z}, \boldsymbol{y})) \big] \Big] \tag{5}$$

$$\min_G \mathcal{L}_G = \mathcal{L}_G^{(\text{sup})} - \alpha \Big[\mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}), \boldsymbol{y} \sim p(\boldsymbol{y})} \log \big[D(G(\boldsymbol{z}, \boldsymbol{y})) \big] \Big], \tag{6}$$

where $\mathcal{L}_D^{(\text{sup})}$ and $\mathcal{L}_G^{(\text{sup})}$ are their respective terms in Equation 1, p_u denotes some unlabelled distribution (i.e. the validation set but with the labels ignored), and α controls the strength of the unsupervised part of the objective. Note that here the generator is not modified to perform unconditional generation – it simply has to fool both the conditional branch $d(\boldsymbol{x}, \boldsymbol{y})$ and unconditional branch $d(\boldsymbol{x})$. α should be carefully tuned here, as if it is too large then the generator may fail to generate class-consistent samples. We are not aware of any work which adapts the projection discriminator in this way, though Sricharan et al. (2017) proposed a semi-supervised CGAN variant which is conceptually similar to Equation 3. To train with this objective, we can simply consider fine-tuning using the entire validation set $\mathcal{D}_{\text{valid}}$ (without labels) for the unsupervised terms in Equation 5, in addition to the labelled support set $\mathcal{D}_{\text{valid}}^{(k)}$ for the supervised terms. In Figure 6a we show the distribution of FID scores obtained when we employ our semi-supervised variant, and also when we are able to tune α. We can see that the latter provides a very significant improvement in FID scores,

and this improvement in sample quality is also reflected in Table 1, which achieves the best mean performance across the board.

Earlier, we explained that mismatches between the generative and real distribution can manifest as degraded accuracy as the number of generated samples increases, and we corroborated this with in Figure 5. To gain further insight into what is going on, we also examine precision and recall metrics (Sajjadi et al., 2018; Kynkäänniemi et al., 2019). Intuitively, precision and recall can be thought of as sample quality and sample coverage, respectively. In Figure 6b we can see that for each set of experiments the average recall lags behind precision; since recall intuitively corresponds to mode coverage this suggests that few-shot generalisation performance is mostly bottlenecked by the lack of sample diversity. (A similar observation can also be made for Omniglot in Figure S8.)



(a) (b)

Figure 6: **Left:** Comparison of FID scores on $\mathcal{D}_{\text{valid}}$ between supervised (Equation 1) and semi-supervised variant (Equation 5). 'Tune α' refers to also tuning the α coefficient. **Right:** Boxplots showing precision-recall metric (proposed in Kynkäänniemi et al. (2019)) over all k's as well as their semi-supervised variants for EMNIST. We can see that the semi-supervised formulations greatly improve both metrics for each k.

In general, we consider generative modelling in the few-shot regime to be a difficult task. All generative models present a trade-off (Xiao et al., 2021) between three criteria: sample quality, fast sampling, and mode coverage/diversity, with GANs tending to be problematic in the latter (Li & Malik, 2018) since they do not perform maximum likelihood estimation (which favours mode coverage). Variational autoencoders (VAEs) on the other hand suffer from reduced sample quality but at the benefit of mode coverage. Score-based (Song et al., 2020) and denoising diffusion probabilistic models (DDPMs) (Sohl-Dickstein et al., 2015; Ho et al., 2020) appear to perform well in terms of both mode coverage and sample quality, and may be a more appealing alternative since its main bottleneck – sample generation – is only of concern if one wants to be able to generate samples in real-time, which is not a concern for us here. Furthermore, as shown in (Song et al., 2020) one can also estimate the log density $\log p(\boldsymbol{x})$ exactly, which avoids the need to use approximate metrics such as FID. Of course, there is still the issue of training or fine-tuning a generative model on limited amounts of labelled data, but we believe a reasonable approach for the time being is through a semi-supervised approach, since unlabelled data is usually abundant. Lastly, we note that it may be an interesting avenue to instead perform model selection using a weighted combination of precision and recall, since this allows us to weight our preferences for sample coverage and quality with regards to model selection.

6 Conclusion

In this paper, we explored the use of conditional generative adversarial networks to perform few-shot data augmentation in order to improve classification performance on various datasets. In general, we found that it is difficult to improve few-shot classification accuracy with a large number of GAN-generated examples due to the inherent mismatch between the ground truth vs generative distribution for the target classes, which leads to overfitting. Part of this is due to GAN intricacies (i.e. the inductive bias to sacrifice mode coverage for sample quality) but this mismatch is heavily exacerbated when there is only k labelled examples per target class. Furthermore, we found that classification performance is highly dependent on how classes are partitioned and this can induce significant variance. To address the difficulty of fine-tuning a GAN to perform k-shot generation, we proposed a pragmatic semi-supervised setup that allows for unlabelled examples to be used for GAN fine-tuning and demonstrated gains in fine-tuning accuracy, FID, as well as precision and recall. The variance exhibited between different dataset seeds is of concern, because it reduces the practical significance of some of the findings. For future work, we suggest building on top of semi-supervised training with more stable and recent generative model classes, such as score-based or diffusion models.

7 ACKNOWLEDGEMENTS

This work was supported by the IVADO Doctoral Excellence Scholarship (PhD-2019b-5593222178).

REFERENCES

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Sergey Bartunov and Dmitry Vetrov. Few-shot generative modelling with generative matching networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 670–678. PMLR, 2018.

Christopher Beckham, Sina Honari, Vikas Verma, Alex M Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Chris Pal. On adversarial mixup resynthesis. *Advances in Neural Information Processing Systems*, 32:4346–4357, 2019.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018.

Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pp. 523–531, 2016.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Arantxa Casanova, Marlène Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. Instance-conditioned gan. *Advances in Neural Information Processing Systems*, 34, 2021.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *arXiv preprint arXiv:1606.03657*, 2016.

Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8680–8689, 2019.

Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*, 2019.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 2525–2534. PMLR, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3018–3027, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. URL http://arxiv.org/abs/1706.08500.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Yan Hong, Li Niu, Jianfu Zhang, and Liqing Zhang. Matchinggan: Matching-based few-shot image generation. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6. IEEE, 2020a.

Yan Hong, Li Niu, Jianfu Zhang, Weijie Zhao, Chen Fu, and Liqing Zhang. F2gan: Fusing-and-filling gan for few-shot image generation. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 2535–2543, 2020b.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019.

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

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13470–13479, 2020.

Ke Li and Jitendra Malik. On the implicit assumptions of gans. *arXiv preprint arXiv:1811.12402*, 2018.

Qi Li, Long Mai, and Anh Nguyen. Improving sample diversity of a pre-trained, class-conditional GAN by changing its class embeddings. *CoRR*, abs/1910.04760, 2019. URL http://arxiv.org/abs/1910.04760.

Weixin Liang, Zixuan Liu, and Can Liu. Dawson: A domain adaptive few shot generation framework. *arXiv preprint arXiv:2001.00576*, 2020.

Emile Mathieu, Tom Rainforth, Nana Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In *International Conference on Machine Learning*, pp. 4402–4412. PMLR, 2019.

Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze discriminator: A simple baseline for fine-tuning gans. *CoRR*, abs/2002.10964, 2020. URL https://arxiv.org/abs/2002.10964.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Esther Robb, Wen-Sheng Chu, Abhishek Kumar, and Jia-Bin Huang. Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*, 2020.

Tim Sainburg, Marvin Thielk, Brad Theilman, Benjamin Migliori, and Timothy Gentner. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions. *arXiv preprint arXiv:1807.06650*, 2018.

Mehdi SM Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *Advances in neural information processing systems*, 31, 2018.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Kumar Sricharan, Raja Bala, Matthew Shreve, Hui Ding, Kumar Saketh, and Jin Sun. Semi-supervised conditional gans. *arXiv preprint arXiv:1708.05789*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pp. 270–279. Springer, 2018.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019a.

Vinay Kumar Verma, Dhanajit Brahma, and Piyush Rai. A meta-learning framework for generalized zero-shot learning. *arXiv preprint arXiv:1909.04344*, 2019b.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7278–7286, 2018.

Davis Wertheimer, Omid Poursaeed, and Bharath Hariharan. Augmentation-interpolative autoencoders for unsupervised few-shot image generation. *arXiv preprint arXiv:2011.13026*, 2020.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *CoRR*, abs/2112.07804, 2021. URL https://arxiv.org/abs/2112.07804.

Yasin Yaz, Chuan-Sheng Foo, Stefan Winkler, Kim-Hui Yap, Georgios Piliouras, Vijay Chandrasekhar, et al. The unusual effectiveness of averaging in gan training. In *International Conference on Learning Representations*, 2018.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. *Advances in neural information processing systems*, 31, 2018.

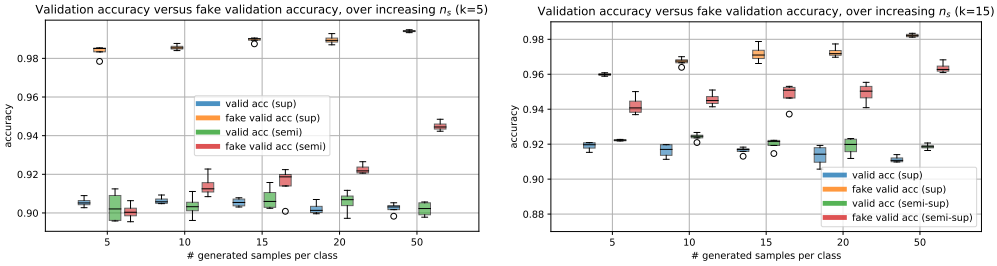
Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

S8 APPENDIX

S8.1 ADDITIONAL RELATED WORK DISCUSSION

One crucial difference between what we propose and both of these works is that ours requires the generative model to be fine-tuned on novel classes. Wertheimer et al. (2020) argue that generative models such as GANs and VAEs have extreme difficulty generalising to novel classes in a 'zero-shot' manner (i.e. without fine-tuning) due to the fact that the training of either models enforces that all plausible regions in the space of the prior distribution decode into plausible samples from the training distribution. They argue that unlike their probabilistic variants, *deterministic* autoencoders are fit for few-shot generalisation because of this lack of rigidity, and propose a novel mixup scheme to ensure interpolations between images in novel classes do not produce images biased towards the training set. We argue that in the case of VAEs however, it depends on the trade-off between the two competing losses that comprise the evidence lower bound (ELBO), which is the likelihood + KL (between posterior and prior). The likelihood (reconstruction error) encourages a bijective mapping between $\mathcal{X}$ and $\mathcal{Z}$, which allows it to generalise to novel inputs. The KL loss acts as a regularisation to make the posterior $q(\boldsymbol{z}|\boldsymbol{x})$ indistinguishable from the prior $p(\boldsymbol{z})$, and in the extreme case would bottleneck the capacity of the network and map many distinguishable inputs to the same mode, which is the opposite to the intended goal of the likelihood term (Esmaeili et al., 2019). An extensive literature surrounds the trade-off between these two terms (Burgess et al., 2018; Esmaeili et al., 2019; Mathieu et al., 2019). A similar analogy could be made for autoencoders with adversarial losses such as F2GAN, where we can imagine the KL loss being replaced with an 'adversarial' divergence between images created by the decoder and images from the training distribution (Berthelot et al., 2018; Sainburg et al., 2018; Beckham et al., 2019). Too much weighting on this term would produce a strong bias in the decoder to ensure *all* images are indistinguishable from those from the training distribution, potentially to the detriment of it being able to generate new samples from inputs coming from unseen classes. This seems to explain why F2GAN contains a multitude of losses to try and encourage sample diversity, though as we have stated, in our work we would prefer to simply fine-tune a GAN to novel classes directly.

S8.2 ADDITIONAL FIGURES



(a) $k = 5$ (b) $k = 15$

Figure S7: Boxplot showing validation accuracy (blue = supervised, green = semi-supervised) and fake validation accuracy (orange = supervised, red = semi-supervised) for different numbers of generated samples per class n_s. The semi-supervised variant is that described in Equation 5. Fake validation accuracy is a held-out validation set generated by the GAN. It can be seen that as n_s increases, the actual validation accuracy (blue) decreases and fake validation accuracy increases (orange), but this effect is mitigated for the corresponding semi-supervised distributions (shown in green for validation accuracy and red for fake validation accuracy). Results are shown over one dataset seed, with each boxplot being a distribution over the different sampling runs.

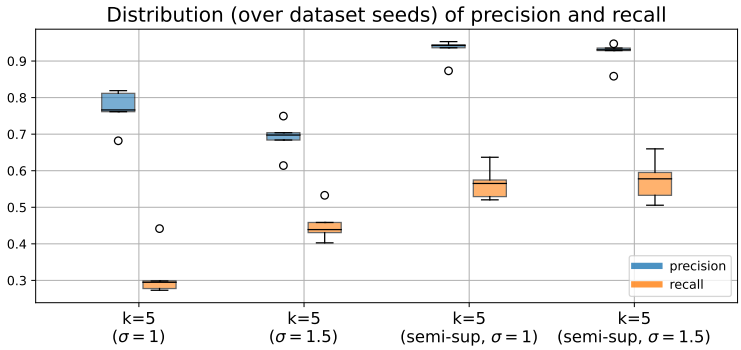


Figure S8: Boxplots showing precision-recall metric (proposed in Kynkäänniemi et al. (2019)) over all k's as well as their semi-supervised variants for Omniglot. Here, we also illustrate the effect of σ, which is the standard deviation of the prior distribution at inference time, i.e. $\boldsymbol{z} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. As demonstrated in BigGAN (Brock et al., 2018), truncating the dynamic range of $\boldsymbol{z}$ leads to increased sample quality at the cost of sample diversity, whereas here we do the opposite by inflating the variance of the prior distribution. We can see that for $\sigma = 1.5$ we obtain better recall (mode coverage) at the expense of precision (sample quality).

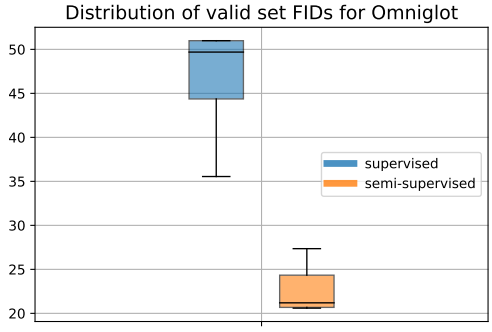


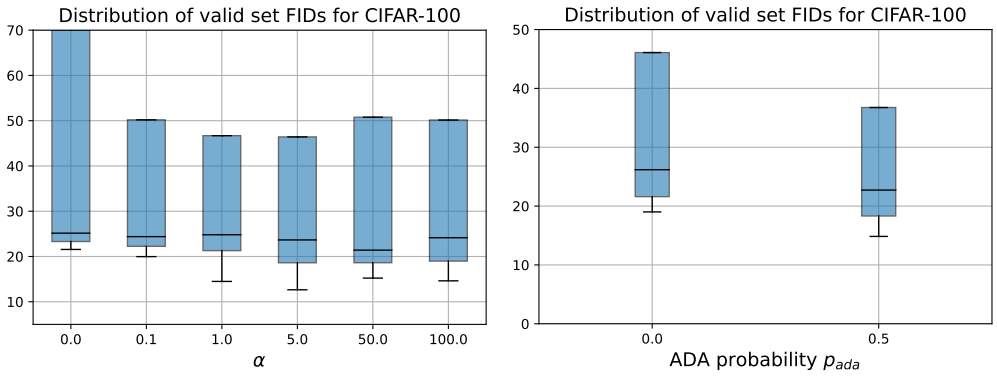
Figure S9: Comparison of valid set FIDs between the supervised and semi-supervised formulations for Omniglot.

S8.3 Additional Results

S8.3.1 CIFAR-100

We further validate our approach on the CIFAR-100 dataset. CIFAR-100 consists of 100 classes of natural images each comprising of 600 images per class. Depending on the dataset seed, we randomly divide the classes into 80% for the source (training) classes and the remaining 20% for the target (testing) classes. Furthermore, we use the official BigGAN (Brock et al., 2018) architecture[3] but modified to be more reminiscent of StyleGAN. Concretely, this involves implementing adaptive batch norm for $\boldsymbol{z}$, replacing $\boldsymbol{z}$-conditioned MLP at the start of the generator with a learnable tensor, implementing exponential moving averages for the weights of the generator (Yaz et al., 2018) and also adding a minibatch discrimination layer after the convolution layers in the discriminator network. In addition to this, we also experiment with the proposed adaptive discriminator augmentation (ADA) trick proposed in Karras et al. (2020) where data augmentation operations stochastically are applied to both the real and generated images without the data augmentation-specific operations 'contaminating' the generative distribution. This is of interest here since such a technique was motivated to be used to improve GANs on limited data. For ADA, we do not use the adaptive variant of the algorithm and simply apply transforms with some probability p_{ada}, where transformations can either be horizontal flips, random rotations, and random translations. In Figure S10 we show the distribution of FIDs with respect to the semi-supervised coefficient α. It can be seen that the best FID is achieved with $\alpha = 5$ (Figure S10a). Using discriminator augmentations is also beneficial to decreasing FID (Figure S10b).

[3] https://github.com/ajbrock/BigGAN-PyTorch



(a) $k = 5$ (b) $k = 15$

Figure S10: For S10a, the distributions are computed over all possible enumerations of *dfm* $\in$ {linear, all}, *gfm* $\in$ {linear, all} and $p_{\text{ada}} \in \{0.0, 0.5\}$. For S10b, the distributions are computed over all possible enumerations of *dfm* $\in$ {linear, all}, *gfm* $\in$ {linear, all} and $\alpha \in \{0.0, 0.1, 1.0, 5.0, 50.0, 100.0\}$.

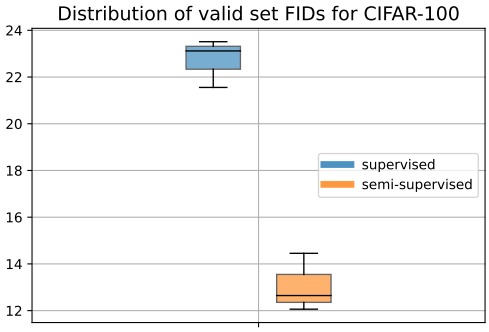


Figure S11: Comparison of valid set FIDs between the supervised and semi-supervised formulations for CIFAR-100.

S8.3.2 EMNIST

See Table S5.

S8.4 Optimal hyperparameters for GAN fine-tuning

In order of dataset seeds 0-4:

S8.4.1 EMNIST k=5

- *dfm* : ['all', 'embed', 'linear', 'all', 'all']
- *gfm* : ['embed', 'embed', 'embed', 'embed', 'embed']
- γ: [100.0, 100.0, 100.0, 100.0, 100.0]

S8.4.2 EMNIST k=10

- *dfm* : ['all', 'all', 'all', 'all', 'all']
- *gfm* : ['embed', 'linear', 'linear', 'embed', 'embed']
- γ: [100.0, 100.0, 100.0, 100.0, 100.0]

S8.4.3 EMNIST k=15

- *dfm* : ['all', 'linear', 'all', 'linear', 'all']

Table S4: Results for CIFAR100, for $k = 50$. Results are averages over three different dataset seeds.

	Method	$\mathcal{D}_{\text{valid}}$	$\mathcal{D}_{\text{test}}$
ours	Baseline w/ data aug	73.55 $\pm$ 1.58	70.78 $\pm$ 2.65
	Fine-tuned GAN	73.46 $\pm$ 1.46	70.79 $\pm$ 3.08
	+ semi-supervised (Sec. 5)	73.56 $\pm$ 1.56	**71.06** $\pm$ **2.31**

Table S5: EMNIST results for $k = 25$ on both $\mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$. For each experiment and its dataset seed, the hyperparameters used for the test set evaluation correspond to the optimal hyperparameters found for the same corresponding experiment / seed on the validation set (e.g. # of epochs, α for input mixup, n_s for GAN).

	Method	$\mathcal{D}_{\text{valid}}$	$\mathcal{D}_{\text{test}}$
ours	Baseline w/ data aug	93.88 $\pm$ 3.75	93.39 $\pm$ 3.99
	Fine-tuned GAN	93.85 $\pm$ 3.61	93.29 $\pm$ 3.86
	+ semi-supervised (Sec. 5)	**94.10** $\pm$ **3.53**	**93.62** $\pm$ **3.80**

- *gfm* : ['embed', 'embed', 'linear', 'embed', 'embed']
- γ: [100.0, 100.0, 100.0, 100.0, 100.0]

S8.5 Optimal hyperparameters for GAN fine-tuning (semi-supervised)

In order of dataset seeds 0-4:

S8.5.1 Omniglot k=5

- *dfm* : ['all', 'all', 'all', 'all']
- *gfm* : ['linear', 'linear', 'linear', 'linear']
- γ: [10.0, 10.0, 20.0, 20.0],
- max epochs trained for: ['215 $\times$ 100', '645 $\times$ 100', '520 $\times$ 100', '354 $\times$ 100']

S8.6 Optimal hyperparameters for classifier fine-tuning

In the order of dataset seeds 0-4:

S8.6.1 EMNIST k=5

- **Baseline**: minScale = [0.6, 0.2, 0.8, 0.6, 0.8], 0.60 $\pm$ 0.22
- **Input mixup**: α = [0.1, 0.1, 0.1, 0.1, 0.1], 0.10 $\pm$ 0.00
- **Fine-tuned GAN**: n_s = [15, 5, 5, 10, 20], 11.00 $\pm$ 5.83
- **Semi-supervised**: n_s = [50, 10, 50, 20, 10], 28.00 $\pm$ 18.33

S8.6.2 EMNIST k=10

- **Baseline**: minScale = [0.8, 0.6, 0.8, 0.4, 0.6], 0.64 $\pm$ 0.15
- **Input mixup**: α = [0.1, 0.1, 0.1, 0.1, 0.1], 0.10 $\pm$ 0.00
- **Fine-tuned GAN**: n_s = [5, 10, 10, 20, 15], 12.00 $\pm$ 5.10
- **Semi-supervised**: n_s = [10, 10, 20, 50, 50], 28.00 $\pm$ 18.33

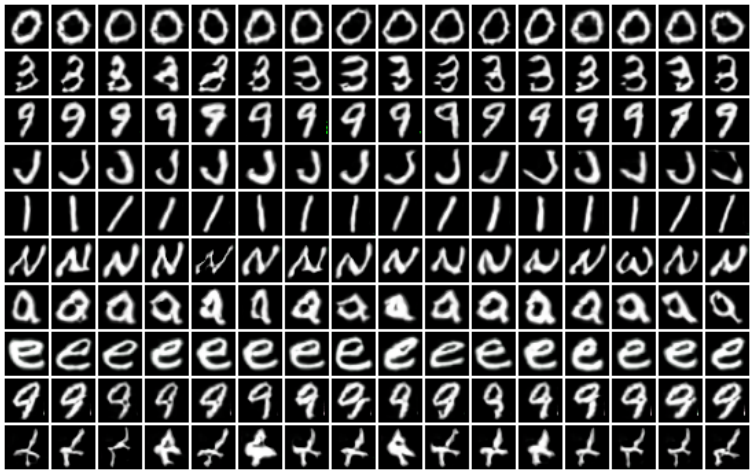
S8.6.3 EMNIST k=15

- **Baseline**: minScale = [0.8, 0.4, 0.8, 0.8, 0.4], 0.64 $\pm$ 0.20
- **Input mixup**: α = [0.1, 0.1, 0.1, 0.1, 0.1], 0.10 $\pm$ 0.00
- **Fine-tuned GAN**: n_s = [5, 5, 10, 10, 20], 10.00 $\pm$ 5.48
- **Semi-supervised**: n_s = [20, 20, 20, 20, 50], 26.00 $\pm$ 12.00

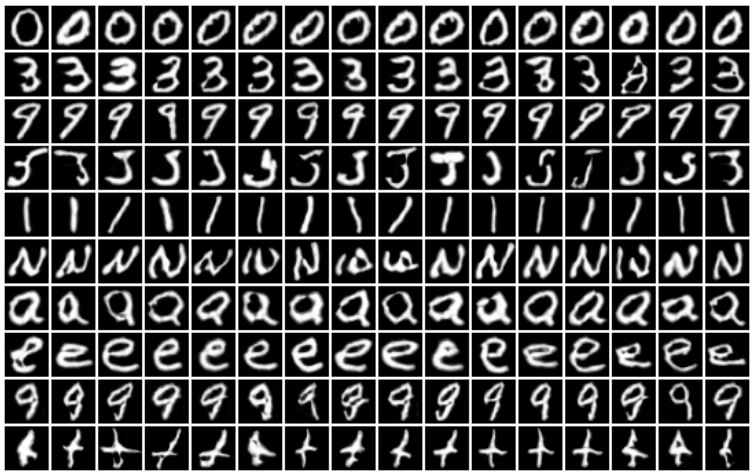
S8.6.4 OMNIGLOT K=5

- **Baseline**: minScale = [0.8, 0.8, 1.0, 1.0, 0.4]
- **Input mixup**: α = [0.1, 0.1, 0.1, 0.1, 0.1]
- **Fine-tuned GAN**: n_s = [1, 1, 1, 3, 2]
- **Semi-supervised**: n_s = [1, 1, 1, 1, 2]

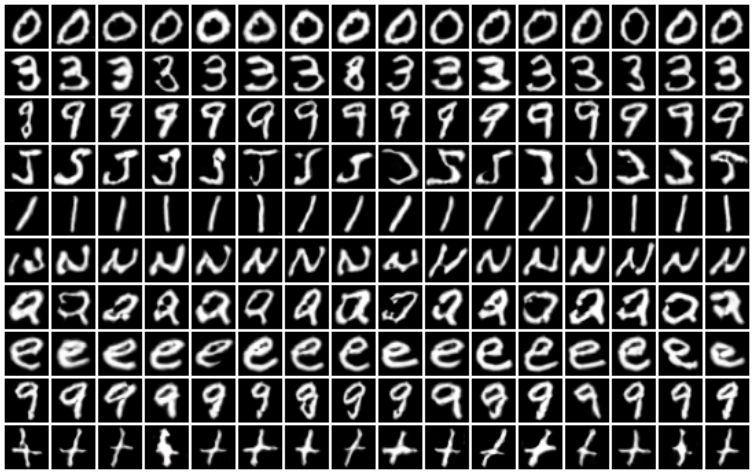
S8.7 GENERATED IMAGES FOR EMNIST (NOVEL CLASSES)



(a) $k = 5$



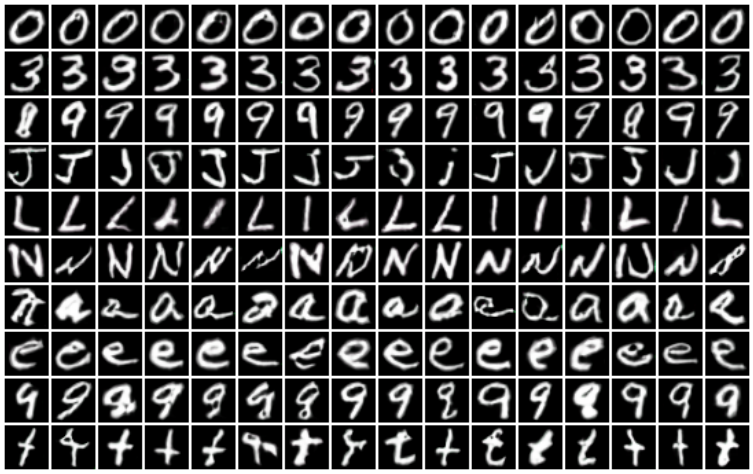
(b) $k = 10$



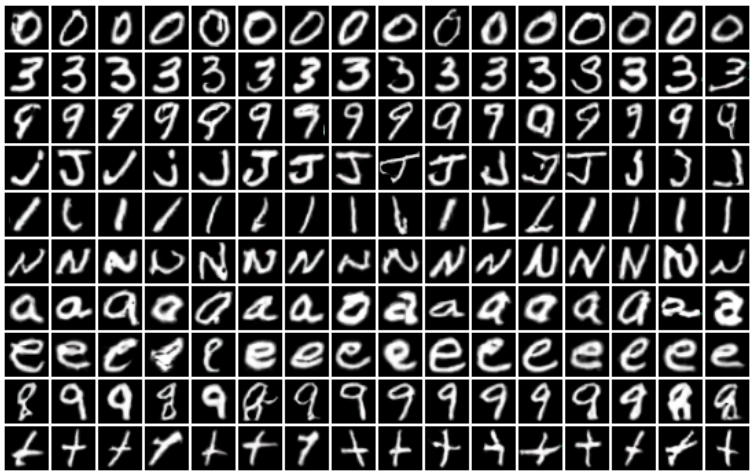
(c) $k = 15$

Figure S12: Generated samples from fine-tuned GAN in Section 2.3.

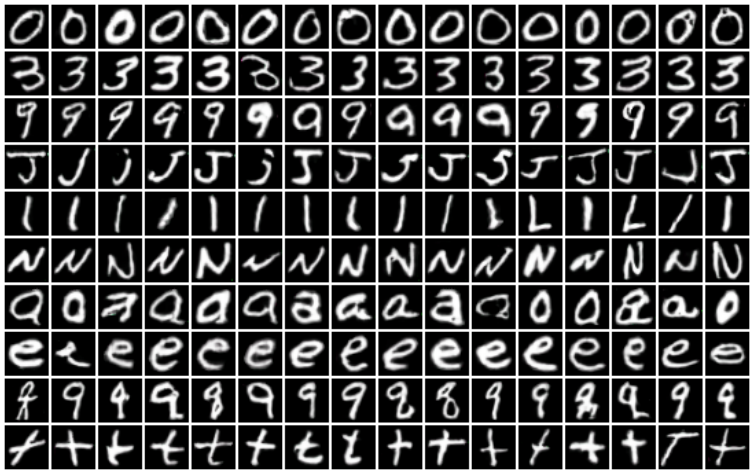
S8.8 GENERATED IMAGES FOR EMNIST (NOVEL CLASSES, SEMI-SUPERVISED)



(a) $k = 5$



(b) $k = 10$



(c) $k = 15$

Figure S13: Generated samples using the semi-supervised formulation in Section 5.

S8.9 GENERATED IMAGES FOR OMNIGLOT (NOVEL CLASSES)

Figure S14: $k = 5$. Only a subsample of classes are shown here due to space constraints.

S8.10 GENERATED IMAGES FOR OMNIGLOT (NOVEL CLASSES, SEMI-SUPERVISED)

Figure S15: $k = 5$. Only a subsample of classes are shown here due to space constraints.

S8.11 GENERATED IMAGES FOR CIFAR100 (NOVEL CLASSES)

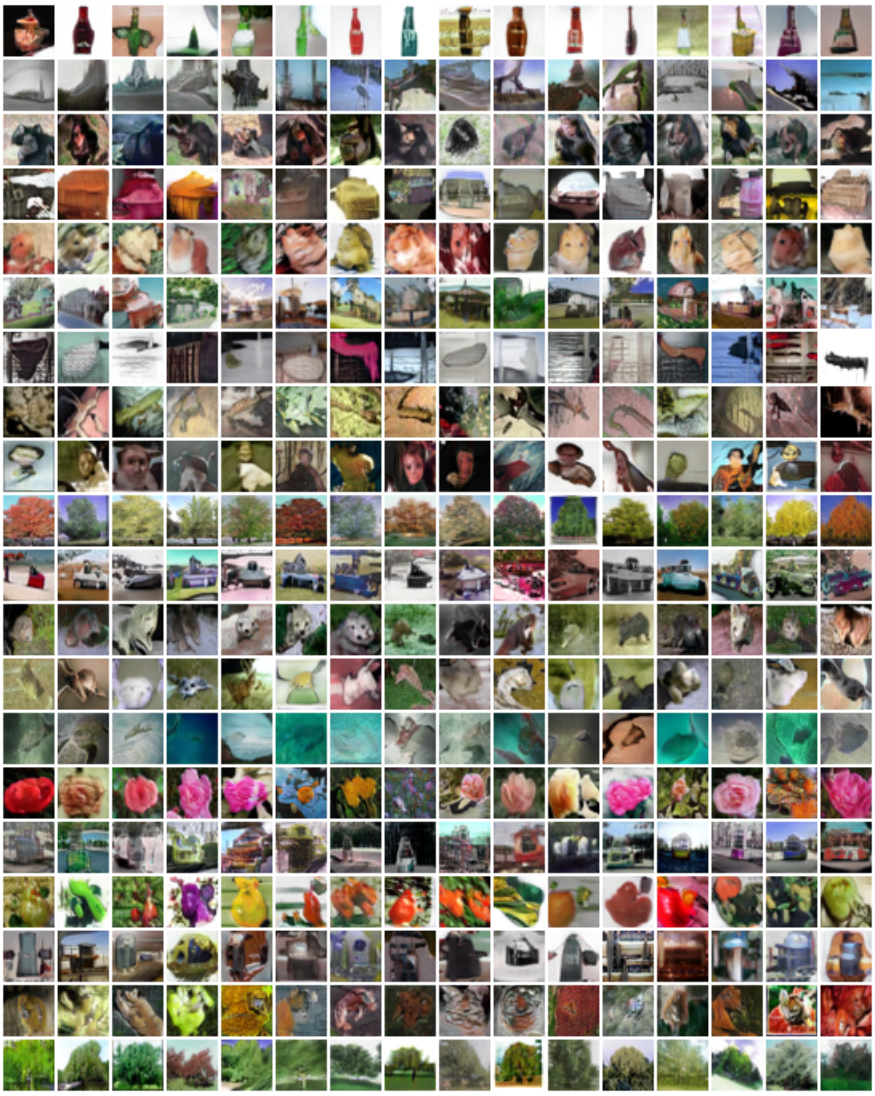


Figure S16: $k = 50$.

S8.12 GENERATED IMAGES FOR CIFAR100 (NOVEL CLASSES, SEMI-SUPERVISED)

Figure S17: $k = 50$.

S8.13 DATASET SPLITS

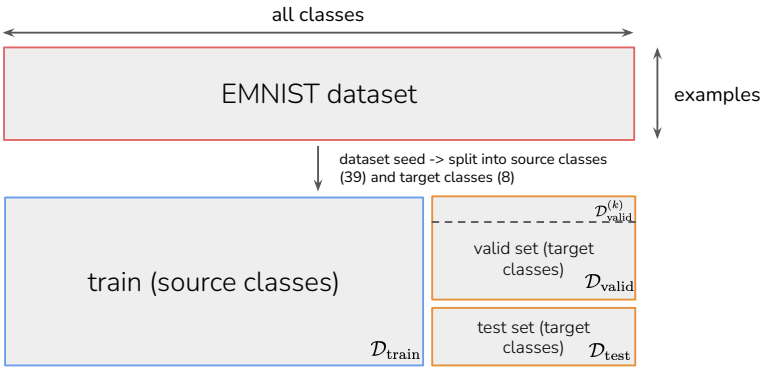


Figure S18: Illustration of how we split our dataset. The dataset seed is used to control what classes comprise training (source) classes and validation/test (target) classes. The GAN and classifier are trained on $\mathcal{D}_{\text{train}}$. The GAN is then fine-tuned on $\mathcal{D}^{(k)}_{\text{valid}}$ and is used to generate examples for the target classes. The classifier pretrained on $\mathcal{D}_{\text{train}}$ is now fine-tuned on $\mathcal{D}^{(k)}_{\text{valid}}$ and the GAN-generated examples and the goal is to maximise performance on $\mathcal{D}_{\text{valid}}$, which is done through hyperparameter tuning. When this is completed, classifier performance is evaluated on the held-out test set $\mathcal{D}_{\text{test}}$.

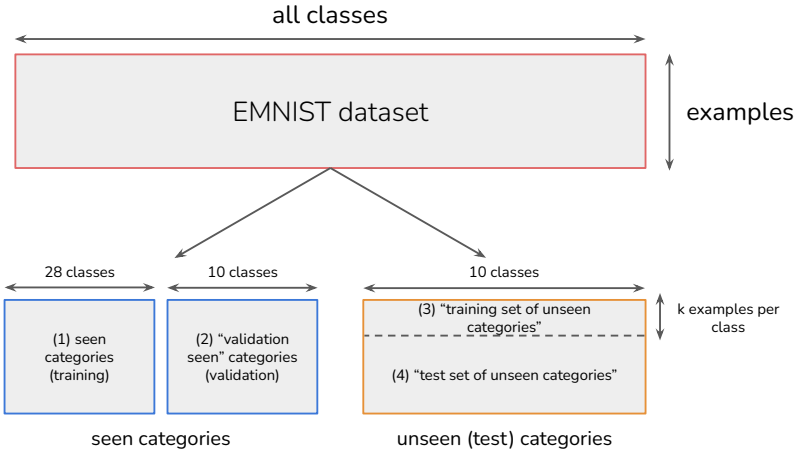


Figure S19: Illustration of how the dataset is split according to MatchingGAN (Hong et al., 2020a). The training set (1) is used to train the GAN, and validation set (2) is used to monitor GAN training (tune GAN training hyperparameters via FID?). What we call the 'support set' they have called the 'training set of unseen categories' (3), and this is what the GAN uses to generate additional examples. The classifier pre-trained on (1) is now fine-tuned on (3) + the GAN-generated examples and is used to predict accuracy on (4). As we mentioned in Section 4, hyperparameter tuning of the generation process / classifier can lead to biased estimates of performance on (4), since there is no additional held-out test set. Furthermore, in F2GAN (Hong et al., 2020b) no validation set appears to be mentioned.

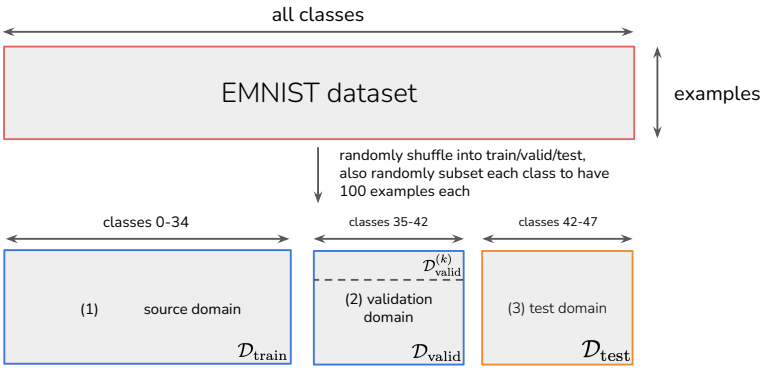


Figure S20: Illustration of how the dataset is split according to DAGAN (Antoniou et al., 2017). Some details regarding the evaluation were not clear, but from what can be gathered, DAGAN is trained on the source domain (training set). A baseline classifier is also trained (from scratch, not fine-tuned) on the validation domain, and hyperparameter tuning is done for n_s to determine how many DAGAN-generated samples per class are optimal. The final evaluation is then training a classifier from scratch on the target domain with the optimal n_s found via hyperparameter tuning. The test accuracy is averaged over five independent runs (dataset seeds).

S8.14 Semi-supervised experiments

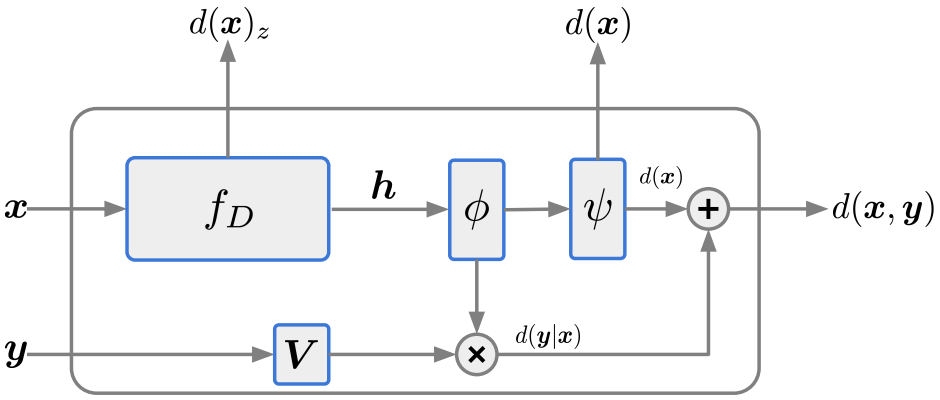


Figure S21: Projection discriminator from Miyato & Koyama (2018). Recall from Section 5 that the output of the discriminator is $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}|\boldsymbol{x}) + d(\boldsymbol{x}) = \boldsymbol{y}^T \boldsymbol{V} \cdot \phi(f_D(\boldsymbol{x})) + \psi(\phi(f_D(\boldsymbol{x})))$. The fact that $d(\boldsymbol{x}, \boldsymbol{y})$ partly decomposes into $d(\boldsymbol{x})$ means that the projection discriminator can be leveraged in a semi-supervised manner.