MeGAN: Meta-Learned Data Augmentation GAN for Few-Shot Image Classification

Mason Swofford
Computer Science
Stanford University
Stanford, CA 94305
mswoff@cs.stanford.edu

John Peruzzi
Computer Science
Stanford University
Stanford, CA 94305
jperuzzi@cs.stanford.edu

Nikita Demir
Computer Science
Stanford University
Stanford, CA 94305
nikita@cs.stanford.edu

Abstract

We attempt to improve few-shot learning for image classification by incorporating data augmentation in which the additional data is generated via a neural network. Low data regimes exist in medicine and robotics and require specially designed inference algorithms. We experiment with multiple approaches which fall under two general algorithms: 1) Using generated images to augment the training dataset and feeding the new dataset to two separate classification and discrimination networks or, 2) Using generated images to improve the decision boundary of a single network with a combined classification and discrimination output. With both approaches we experimented with different forms of generator conditioning: no conditioning, task labels, or input images; we also experimented with a variety of loss functions for the inner and outer loop MAML updates that varied the impact of the generated images on standard MAML learning. Our results show faster convergence and higher accuracy compared to the standard MAML approach on the Omniglot dataset.

1 Introduction

1.1 Machine Learning in Low Data Regimes

Deep learning naturally is a data hungry process, and a substantial portion of modern deep learning success can be attributed to the explosion of massive datasets and development of techniques for handling them. However, despite this progress, there still exist some industries, such as medicine and robotics, where datasets are fundamentally limited in size, because collection and labeling is dangerous, expensive, and/or time-intensive. These fields require deep learning techniques that can learn to make decisions from very few input examples. Meta-learning is one class of techniques for learning in low data regimes.

1.2 Meta-Learning Preliminaries

Meta-learning within the context of Machine Learning is a sub-field that, informally, is concerned with "learning how to learn." Often, it is applied to K-shot N-way classification problems, where the goal is to learn a classifier that can flexibly and easily generalize to new tasks given only K (few) training examples for each of N classes. The model trains and tests on tasks T_i drawn from a distribution of tasks $p(T)$, during a phase called *meta-training* before final evaluation, *meta-testing*, on held-out tasks.

Our approach to this problem utilizes the Model-Agnostic Meta-Learning (MAML) algorithm [4]. MAML is used to learn a model $f_\theta : X \rightarrow Y$. Under MAML, given a new task T_i with K training

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

examples for each of the N classes, f begins with a set of initial parameters θ, and then performs gradient steps to optimize θ to θ_i' according to the loss function L and the $K \times N$ training examples. Then, $f_{\theta_i'}$ is used to classify the test examples for this task. The initial parameters θ are then updated by gradient descent using higher-order gradients optained by applying the loss function L_{meta} to $f_{\theta_i'}$'s test outputs and differentiating through the training process with respect to the initial θ. This process is repeated throughout meta-training, on batches of tasks $T_i \sim p(T)$ as shown in Figure 1. We chose to use MAML for our meta-learning procedure, because it's higher-order update is agnostic to which differentiable model it is updating. This gave us versatility in architecture designs.

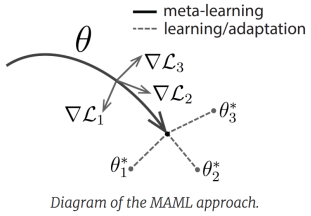

Figure 1: MAML update procedure. θ_i^* are optimized for specific tasks with initial weights θ. θ is updated in the meta-update after each batch of tasks

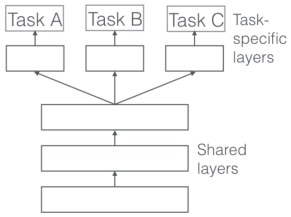

Figure 2: An example multi-task neural network architecture in which some weights are shared before the networks diverge.

1.3 Multi-Task Learning Preliminaries

Multi-task learning is a sub-field of Machine Learning in which distinct tasks are solved simultaneously, with the goal of exploiting commonalities and differences across tasks. To do this, neural network architectures often share some features between tasks, while also maintaining task-specific features (see Figure 2). It has been shown, in some cases, to improve task performance compared to when task-specific models are trained separately [2]. It is thought to be especially effective when similarities are shared across tasks [6]. Some have framed it as a superior form of regularization since it does not penalize all complexity uniformly [3]. Based on these insights, we believed we could improve our model's N-way classification accuracy by incorporating a generative network and a second task: real/fake discrimination.

1.4 Conditional GANs

Conditional GANs [9] are a type of Generative Adversarial Network in which a class-conditioning vector is appended to the input of the generator and discriminator, causing the generator to learn class-conditional distributions. The class-conditioning can take many forms, such as the true label y [9] or a class-aggregated feature prototype [7]. We experimented with both of these conditionings as well as passing in raw images.

1.5 Related Works

There has been some prior work on using GANs for data augmentation. [14] gets summary statistics from the training images and then uses a GAN to produce fake data. Their approach takes an existing K-shot N-way classifier and adds an additional "fake" classification category. They feed the classifier both real and fake images, and attempt to distinguish between the now N+1 categories. This method treats the GAN inputs as unsupervised data, which loses information contained in the class labels. The N+1 classification objective can be viewed as a multi-task learning objective, combining the tasks of discrimination and classification, with fully shared weights, as every layer in the model assists with both classification and discrimination.

[13] attempts to produce augmented versions of each training image by using a generator. The generated images are then appended to the original training data, and a classifier is trained on top of this augmented dataset. The loss of the classifier is back-propagated through the generator optimizing it for the classification task and not for producing realistic images. Because their GAN is only

Figure 3: Omniglot Dataset Examples. Top-left and top-right are characters from the same class.

leveraging information from one image at a time, this method reduces to a classifier with some random inputs.

[11] uses numerical data, and shows similar results to SMOTE or ADASYN, with the surprising result that models trained exclusively on generated data could outperform models trained on the original data. The clear distinction between our work and theirs is that we plan to use image data, which is where GANs have shown the most promise over prior statistical methods.

2 Problem Statement

2.1 Dataset

The two most prominent datasets for few-shot image classification are Omniglot [8] and Mini-ImageNet [12]. Omniglot is composed of 28×28 black and white images of characters from various alphabets, and tasks correspond to different alphabets (Figure 3). Mini-ImageNet is composed of 84×84 color images of various classes of objects, and tasks require classifying a subset of object classes. In order to quickly iterate our experiments and demonstrate their viability, we primarily experimented on the Omniglot dataset. Omniglot is made of 50 alphabets with 30 and 20 classes for meta-training and meta-evaluation, respectively. There are a total of 1623 individual characters from real and fictitious languages. We also implemented our work on the Mini-ImageNet dataset; however, due to its complexity and size, models took longer to train, and we opted to primarily use Omniglot.

2.2 Problem Formulation

In our problem we have a distribution of related tasks $T_i \sim p(T)$. The tasks are to predict a label Y, given an input X. We attempt to meta-train on a set of labeled tasks so that during meta-test, when presented with a new task T_{test}, we can efficiently learn an effective mapping $f : X_{test} \rightarrow Y_{test}$. In our specific experiments, X, Y are images and their respective classes. For each task, our problem is made difficult as we only have access to K (few) examples of each of N classes. The individual classification tasks T_i are split into meta-training and meta-testing groups (60/40 split). During meta-training the networks update their general parameters between tasks and then for a specific task these are then fine-tuned during the inner update loop. During meta-testing the network is given a new task and it quickly optimizes to make a few-shot prediction on meta-testing task's test set.

3 Technical Approach

At a high level, we can segment our approaches into two general frameworks: 1) Separating our classification and discrimination networks in order to output both a classification and real/fake prediction or, 2) Combining the classification and discrimination networks into one network, that outputs a prediction that the input is **either** one of N classes **or** generated by the generator. The latter approach was inspired by [14] and proved to be more successful than the former. We experimented heavily with different loss functions between the inner and outer MAML update steps as well as a variety of ways to architect the generator.

3.1 Generator Configuration

While we experimented with various approaches, one of our generator architectures can be seen in the top of Fig. 4. We tried several different conditioning approaches: 1) Conditioning on an average pooled embedding from passing every real image of a particular class through a pre-trained ResNet-18 and extracting a feature embedding, 2) conditioning on a real image itself and, 3) conditioning on

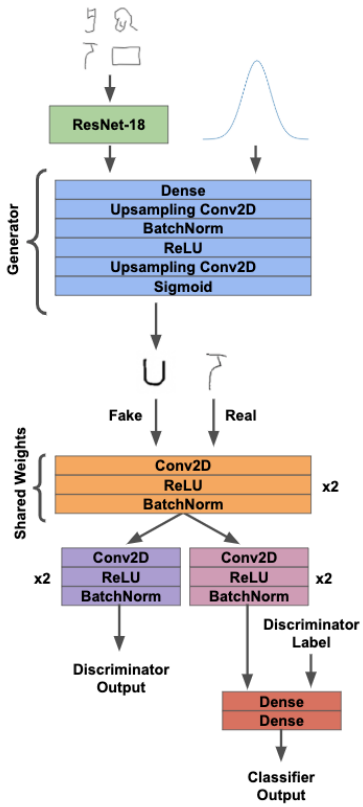

Figure 4: Our multi-task architecture

a real image's one-hot encoded label. To make generating realistic images easier, we also tried outputting the residual on top of a real image. That is, given our generator G and conditioning on a real image x, we would output $G(x) + x$.

3.2 Multi-Task Classification and Discrimination

To incorporate our GAN architecture into the MAML learning procedure, we originally formulated our objective as a classification task and a discrimination task and solved it using two networks with some possibly shared layers. Generated and real images would be input into the combined network, and it would output both a predicted classification and real/fake discrimination. Fig. 4 shows an example of this architecture. We experimented with varying amounts of layer sharing, ranging from two completely separate networks, to sharing all weights until the final layer. Our intuition for sharing some weights was that the discrimination and classification tasks shared similar underlying structure, in that the tasks both require understanding the basic structure of an input image. We saw this implementation as a natural way to extend MAML with a generative network without disrupting its normal training procedure. At training time, we generated a few samples conditioned on the training data and appended them to the training set. The classifier and discriminator then outputted their predictions and a gradient was taken with respect to their individual losses. During the testing step, we calculated just the classifier's loss on the real test examples, using the weights updated during training. This loss could then be unrolled to update the classifier, generator, and discriminator between meta-training steps. We hypothesized that this approach would lead the generator to produce images that helped the classification at test time, which is what we ultimately care about, while still having to fool the discriminator during training time. A key aspect of our multi-task approach was that our classifier was conditioned on the true real/fake value of each input image, which allowed it to process real and fake data differently, if it was helpful. We hypothesized that this would allow the standard MAML Omniglot classifier to remain mostly unchanged when seeing real data but to selectively incorporate information from its classification on the fake examples.

3.3 N + 1 Classification

We also explored an additional architecture formulation in which our classifier and discriminator were one network that could output either an N-way classification or an extra label corresponding to fake images. We were inspired to take this approach by [14] and their rationale that this single task helps "sharpen the decision boundary" of the classifier during test time. Unlike the other approaches, we did not condition this network on whether an input image was real or fake, and therefore the distinction had to entirely be learned.

3.4 Loss Functions

We experimented with a number of different loss functions for the training and meta-training updates. The classification loss was always the cross-entropy loss. The disrimination loss was either the vanilla GAN loss, the non-saturating GAN loss, the Wasserstein GAN loss, or the Wasserstain-GP GAN loss. While we started with the vanilla GAN loss and non-saturating GAN loss, we found our model did not learn as well as we had hoped.

To encourage our generator's convergence and avoid mode collapse, we examined Wasserstein GANs, which use a slightly different loss function in addition to gradient clipping for improved results and stability [1]. This also has the additional benefit of reflecting image quality so we can tell the learning progress from its value rather than having to qualitatively sample images. The critic's new loss function then is $D(x) - D(G(z))$ and the generator's is $D(G(z))$. However, a key requirement is that the weights are clipped. We also experimented with the Wasserstein-GP loss, which is the same as the Wasserstein loss, except with an additional penalty on the norm of the gradient, in place of gradient clipping [5].

We also varied the training loss and meta-training loss. Our base meta-training loss was the cross-entropy loss on real examples. However, we experimented with adding various combinations and weightings of the cross-entropy loss on the generated examples, as well as the discrimination loss. As we expected, adding the discrimination loss to the meta-training update caused the model to produce more realistic generated images, but hurt meta-test test classification.

Algorithm 1: GAN-Augmented Meta-Learning

Require $p(T)$: distribution over tasks;
Require f_θ: Multi-task N-way classifier and real/fake discriminator;
Require G_ϕ: Generator;
Require α, β: step size hyperparamaeters;
while *not done* **do**
  Sample batch of classification tasks $T_i \sim p(T)$;
  for all T_i **do**
    Sample K datapoints $R_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from T_i;
    Set $\theta', \phi' = \theta, \phi$;
    for all *inner-gradient steps* **do**
      From G generate M fake samples $F_i = \{\mathbf{x}^{(k)}, \mathbf{y}^{(k)}\}$;
      Using $D_i = \{R_i, F_i\}$, and losses from 3.4, perform inner-gradient updates:
      $\theta'_{sh} \leftarrow \alpha \nabla_{\theta'_{sh}} \mathcal{L}_{sh}(f_{\theta'})$
      $\theta'_{cls} \leftarrow \alpha \nabla_{\theta'_{cls}} \mathcal{L}_{cls}(f_{\theta'})$
      $\theta'_{dsc} \leftarrow \alpha \nabla_{\theta'_{dsc}} \mathcal{L}_{dsc}(f_{\theta'})$
      $\phi' \leftarrow \alpha \nabla_{\phi'} \mathcal{L}_{gen}(G_{\phi'})$
    end
    Sample datapoints $R'_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from T_i for the meta-updates;
  end
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \Sigma_{T_i \sim p(T)} \mathcal{L}_{cls}(f_{\theta'_i})$ and $\phi \leftarrow \phi - \beta \Sigma_{T_i \sim p(T)} \mathcal{L}_{cls}(G_{\phi'})$ using $D_i = R'_i$
end

4 Results

We ran all of our experiments on the 5-way 1-shot Omniglot image classification task, and we were impressed to report both improved accuracy over standard maml and faster convergence. However, we note that certain optimization and architecture choices made it impossible to achieve good learning.

For our baseline we implemented the standard MAML approach pioneered by [4].

4.1 Multi-task Approach

With our original multi-task approach in Algorithm 1, we found that we were at best able to achieve slightly below baseline performance despite trying multiple iterations of shared weight overlap. We thus ran an ablative study to analyze the components of our architecture individually. An architecture with no weights shared between the discriminator and classifier revealed that the generated examples were hurting the classifier's ability to learn when it optimized for classification of the generated classes as well as the real (see Figure 10). We noticed that in spite of losses such as Wasserstein providing more stable generators, and in spite of meta-updating both the discriminator and generator to improve the classification loss, the examples generated were only harming the classifier. We hypothesized that this was due to the nature of our training loss; even though our generator was meta-optimized to help the classifier, in training where large gradient steps occur, the generator was agnostic to the performance of the classifier, only trying to fool the discriminator. We thus proceeded an adjusted approach: the N+1 Classification which would not confuse the classifier by forcing it to define classes for noisy images.

4.2 N + 1 Classification Approach

We found much better results with our N + 1 classification approach and were impressed by slightly better performance and convergence with our final approach compared to MAML. We found that for our model to train well with the generative network's input we needed to weigh the loss component for the classifier network slightly less on fake examples. In essence our first approach weighted equal and fake examples contributions to the inner loop's classification loss equally since we were generating a fake example for each real example. However, this seemed to provide too much noisy information to the classifier network which inhibited our training which we show in the performances in 5. We ended up setting this weight factor to weigh the real examples' contribution to the loss 3 times more heavily than the fake ones and this helped our model start generalizing between tasks. Furthermore, inspired by the common GAN practice of training the discriminator for more iterations than the generator, we tweaked our inner loop task-level optimization procedure to only start classifying generated images after 3 out of 5 total inner updates had happened. We hypothesize that this has the effect of allowing the classifier to first get itself situated in the new task so that it can then better discriminate the outputs of the generator against the new images. The performance differences were large and we compare them in 5.

We also experimented with different versions of our outer loop loss functions. This outer loop loss is calculated on the test dataset using the classifier weights after multiple inner loop updates. We compare the results of these different outer loop loss structures against baseline MAML in 6. Most notably, we see that the outer loop loss plotted in grey outperforms standard MAML plotted in red. This one corresponded to only caclulating the outer loop loss on real examples and the classifiers performance on them during testing which then is backpropogated to the classifier and to the generator through the rolling out of the inner loop updates. To summarize, our best performing model used a loss for the outer loop update also called the meta-update that only focused on how well the classifier was classifying real examples. We also tried a meta-update loss in which we specifically evaluated the classifier on fake examples generated from augmenting the training dataset as well as the real images from the test set (shown in orange) and a hybrid of these two where we kept the meta-update for the classifier as solely based on the real example in the test set but then updated the meta-parameters of the generator by how well it fooled the classifier (shown in blue). Overall we were excited to see an improvement over MAML, even if slight, because of the tremendous effort that was required to structure our model in a way that we would generalize between meta-updates.

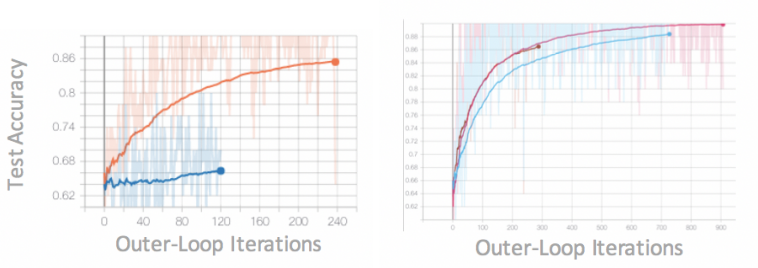

Figure 5

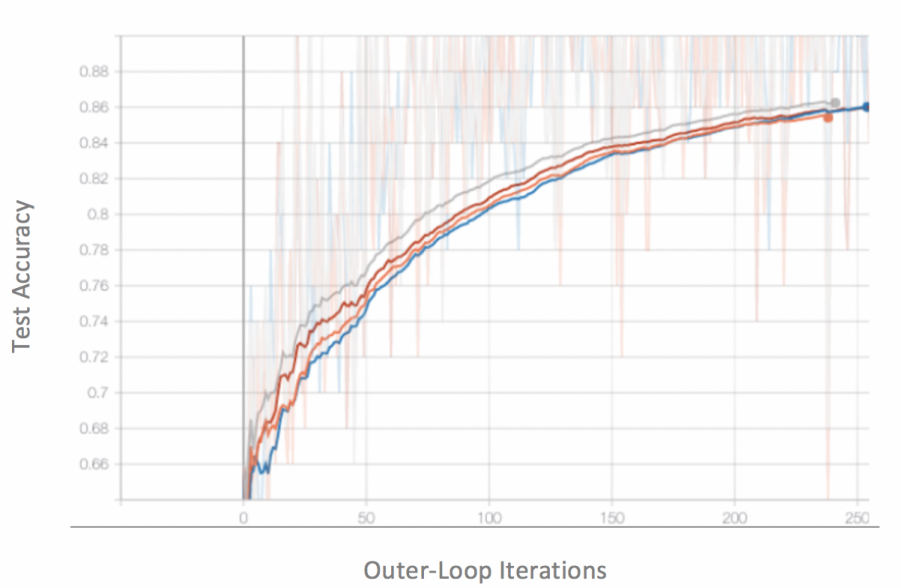

Figure 6: Accuracies with the N+1 Classification Method

We show sampled images from our generator with and without a skip connection in 9. In both cases it seems that the generator is not strictly outputting noise; however, without the skip connection it becomes quite difficult to make out what the generated example represents.

5 Qualitative Analysis

5.1 Sensitivity to the Generator Model

5.1.1 Few-Shot Generation is Non-Trivial

Generative networks are notoriously difficult to train. Our experiments note that this is particularly true in the few-shot setting. Even standard algorithms whose parameters had been finely tuned to produce realistic images begin to fail with $k < 25$. We demonstrate this phenomenon on a standard GAN which generates reasonable images when trained on a large amount of examples from the Fashion-MNIST dataset. (Figure 7) We evaluate this experiment on Fashion-MNIST as the classes have far more examples than Omniglot. With few enough examples the generator often stops generating realistic samples, mode-collapses, outputs noise, or begins to experience numerical instability.

We noticed a similar phenomena with Omniglot during initial training, which negatively impacted a classifier trained on such examples. To amend this, instead of naively applying an out-of-the-box conditional generator, we ran many of our 1-shot experiments with a GAN conditioned on the

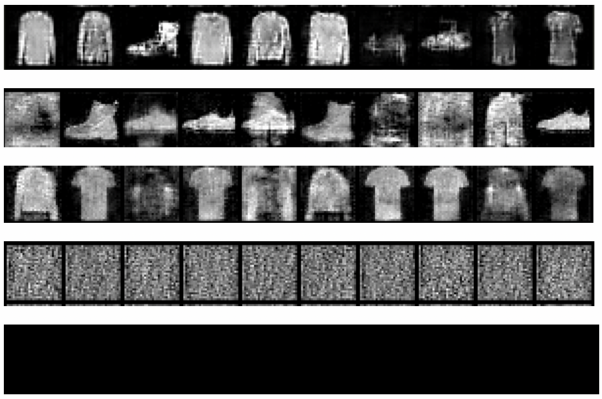

Figure 7: Fashion-MNIST Generated Examples from K training examples after 500 gradient updates using an out-of-the-box GAN. From Top to Bottom: $K = 100, 25, 10, 5, 1$

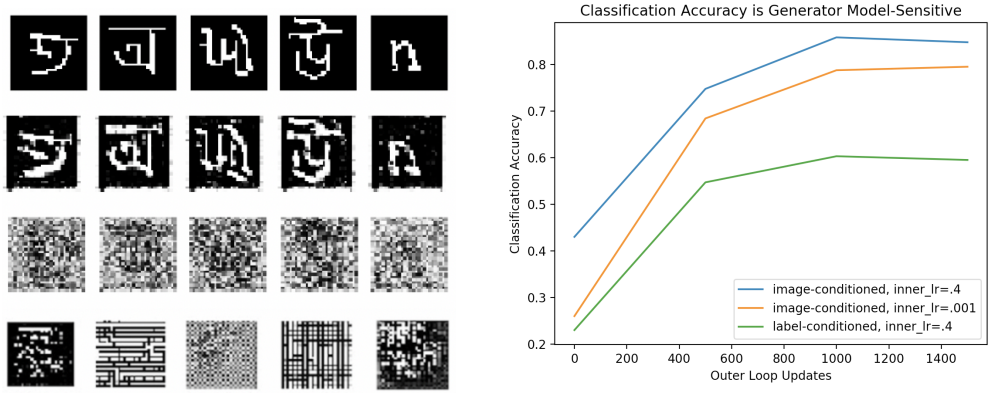

Figure 8: **Left:** Generated Characters after 1000 meta steps. From Top to Bottom: Original Characters; Generated Conditioned on Image (generator inner lr=.4); Generated Conditioned on Image (generator inner lr=.001); Generated Conditioned on Labels (generator inner lr=.4) **Right:** Classification accuracies for models corresponding whose classifier is trained with images on the left.

image itself, making it easier for the generator to learn to output images other than noise and even experimented with adding skip connections.

5.1.2 Study of Sensitivity of Generator and Impact on Classification

A perhaps unsurprising result is that the generator's ability to generate realistic images is highly sensitive to its architecture and parameters. We performed a study on different GAN conditionings and hyperparameters to note its sensitivity to them. Conditioning on the training image instead of labels drastically improved stability and visual realness of the images. A simple change in the inner learning rate from .001 to .4 (from a standard generator to a more standard MAML inner learning rate) gave the generator capacity to produce realistic images. See Figure 8

A more nuanced result of this study is the impact that the capacity of the generator had on the classifier. In this experiment, the classifier shared no weights with the discriminator- we were purely evaluating the image's ability to augment the dataset by serving as additional examples of each class. The generators which had the capacity to generate less noisy images assisted the classifier better than the generators which did not. Figure 8

This is less the case for the N+1 Classifier aporoach (Figure 9), as noisy examples could improve the decision boundary without confusing the classifier into optimizing for classifying noise as legitimate examples.

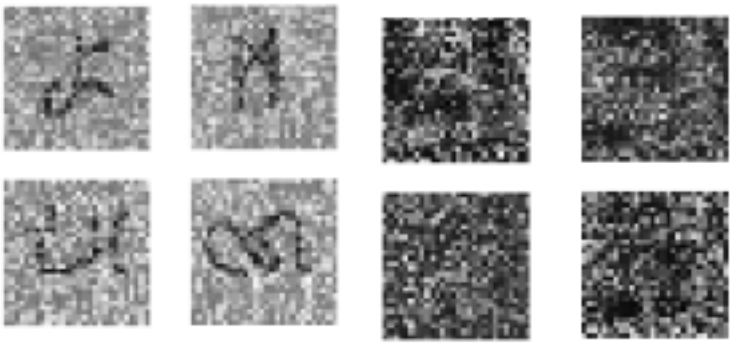

Figure 9: Sampled outputs of our generator on a particular task in the N + 1 way approach. **Left:** With a skip connection FCN-style generator architecture **Right:** Standard generator architecture that led to improved accuracy over MAML

This represents a difficulty of using a generator, whose architectures are difficult to tune and whose errors are are difficult to dissect, to improve classification in the few-shot setting via augmentation of class examples but also the potential for improvements to our results, simply by implementing a more capable generator (a benefit of the model-agnosticity of MAML). Generators in the few-shot setting have shown recent improvements and thus indicate potential improvements for our model as well [10], though many of these few-shot generators are currently quite slow.

6 Conclusion

We implemented a generative network to assist with classification in the few-shot setting. We hypothesized that the generator could assist via augmenting the small dataset, potentially leveraging shared information accross classification and real/fake discrimination tasks and or by improving the decision boundary. One our architectures did improve upon the standard MAML approach, indicating that in the end we were able to utilize a generator to improve classification. The best performing model was the N+1 Classification approach, which shared almost nearly all weights in the discriminative and classification tasks by adding an additional class for fake images. We analyzed some of the failure modes of other approaches and discovered that key to the success of effectively augmenting the actual classes of the images is a generator which can easily learn in the few-shot setting, which we also show is itself not a simple task.

In the future we hope to experiment more with various amounts of weight sharing. In dealing with failure cases of our larger model, we were only able to implement the N+1 approach for joint classification and discrimination. Future experiments should show how splitting these networks earlier improves results.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[3] Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine Learning*, 85(1):149–173, Oct 2011.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[5] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

[6] Ehsan Hajiramezanali, Siamak Zamani Dadaneh, Alireza Karbalayghareh, Mingyuan Zhou, and Xiaoning Qian. Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data, 2018.

[7] Sai Kumar Dwivedi, Vikram Gupta, Rahul Mitra, Shuaib Ahmed, and Arjun Jain. Protogan: Towards few shot learning for action recognition. In *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

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

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[10] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image supplementary material.

[11] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. Data augmentation using gans. *arXiv preprint arXiv:1904.09135*, 2019.

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

[13] 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*, pages 7278–7286, 2018.

[14] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems*, pages 2365–2374, 2018.

7 Appendices

7.1 Ablation Study on Generated Examples

Removing all shared weights from the network we were able to test the hypothesis that a generator trained in the meta-loop to optimize the classifiers update but trained in the inner-loop to fool a discriminator would improve the classifier by augmenting the number of shots of each class. We were unable to train a generator to do so. In Figure 10 is a plot of our optimal generator between all inner-loop loss functions that optimized to fool a discriminator.

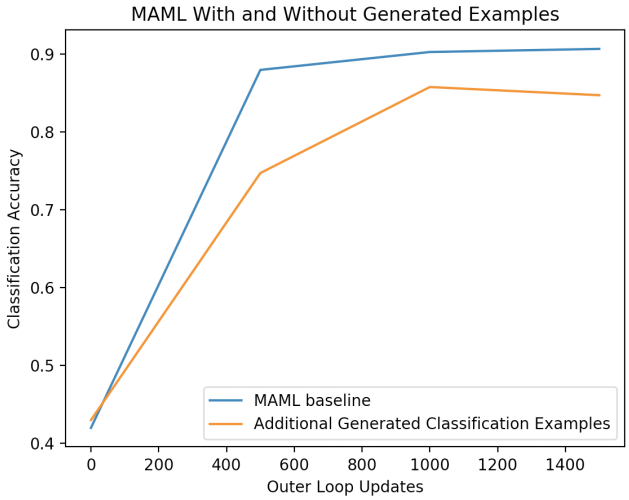

Figure 10: No weight sharing between discriminator and classifier - the generated examples serve only two provide more shots of each class.