FARM: Family of AutoRegressive Models

Mihir Patel
Department of Mathematics
Stanford University
mihirp@stanford.edu

Vinjai Vale
Department of Computer Science
Stanford University
vinjai@stanford.edu

Eric Frankel
Department of Computer Science
Stanford University
ericsf@stanford.edu

Abstract

A major fallacy of autoregressive models is their dependence on ordering. For domains like images, where there are no natural orderings, models like MADE and PixelCNN are intractable in scenarios with partial or reordered evidence despite having success on generative tasks for full images with raster orderings. We propose a new method, FARM, for improved image completion by learning a **F**amily of **A**uto**R**egressive **M**odels with different orderings[1]. We provide various techniques to help FARM converge, including weight sharing and using generalized Jensen-Shannon Divergence (JSD) to force different orderings to yield the same distribution. We demonstrate the potential for this approach in image inpainting, showing FARM outperforms a single ordering. We also show how FARM can be used for fast sub-linear sampling.

1 Introduction

Generative modelling tasks, whereby we wish to generate new content such as images or text, is an important problem space with many recent breakthroughs. Autoregressive models have emerged as a successful approach to the challenge of generative modeling. At their core, autoregressive models approach the problem of understanding a joint distribution by using the chain rule to factor it as the product of *ordered* conditionals. For some ordering, we see can model each subsequent conditional to generate the parameters of an appropriate distribution:

$$p_{\theta_i}\left(x_i|\mathbf{x}_{<i}\right) = \text{Dist}\left(f_i\left(x_1, x_2, \ldots, x_{i-1}\right)\right).$$

With the advent of highly successful deep learning methods which exploit large neural networks, autoregressive models have become considerably more expressive and powerful. In particular, works such as MADE [1] and PixelCNN [2], along with subsequent variants, have proven to be extremely successful. More modern approaches which combine autoregressive models with other techniques have shown state of the art results from Parallel Wavenet [3] for speech synthesis to VQ-VAE-2 [4] for image generation.

However, an important limitation of autoregressive models is $O(n)$ sampling. As each value in the ordering is dependent on all prior ones, we must go through this ordering sequentially. This limitation is considered intrinsic to the approach, and as best as the authors can tell, there have been no successful attempts to mitigate this by investigating ways to trade off sample quality for faster sampling.

[1]The code for this paper can be found at www.github.com/esfrankel/farm.

Preprint. Work in progress.

Additionally, a key fallacy of this approach is its dependence on ordering. In domains such as images, where there is no natural ordering, the standard approach has been to pick some simple contiguous one such as a raster ordering, which starts at the top left and descends down in rows. This becomes problematic when we look at tasks such as image inpainting, otherwise known as image completion, where we wish to fill in a missing part of an image. Normally, we would use Bayes' rule to condition on all the evidence and fill in the missing part using our model. However, due to the number of possible values a pixel can take on, Bayes' rule is intractable and we are only able to use evidence which comes before the point which we want to fill in given our ordering. As an example, if the top half of an image is occluded, a standard raster ordering is unable to use the bottom half to fill in the top. This problem exists in all domains, but becomes especially problematic in images where data has no natural ordering as opposed to text, where we could learn a forward and backward ordering.

To combat these problems, we explore learning FARM, a **F**amily of **A**uto**R**egressive **M**odels. This family of models is able to exploit more evidence than a single ordering for challenges like image infilling, and gives the first autoregressive approach to sub-linear sampling. In this work, we first show various ways in which existing models, namely MADE and PixelCNN++, can be adapted to learn ensembles. In particular, we look at ways to efficiently learn multiple orderings, which is generally intractable as there are $n!$ possibilities, and propose methods to ensure the distributions learned by multiple orderings are similar. We complement this with theoretical results showing a distinct tradeoff between expressivity of each ordering and the cohesiveness of the ensemble. We then provide experimental results on the challenge of image inpainting, showing FARM is able to outperform the current approach of raster ordering. Lastly, we show a way to achieve sub-linear sampling using this family.

To summarize our contributions are:

1. We propose ways to effectively learn FARM on existing autoregressive models, exploring ways *improve* tractability and ensemble cohesiveness.
2. We provide some theoretical insight into our approach, exploring limitations and *tradeoffs* between tractability and ensemble cohesivness.
3. We show adding FARM to existing methods gives better results for image inpainting.
4. We provide the first method for sublinear sampling with minimal performance degradation.

2 Related Works

To the best of the authors' knowledge, this is the first work to explore learning a family of autoregressive models (FARM) with multiple orderings. The same goes for achieving sublinear inference time. For this section, we will focus on the models models which we adapt with FARM along with brief mentions of learning multiple orderings we were able to find in the literature. We will also discuss the image inpainting problem and sublinear sampling.

2.1 MADE

Masked Autoencoder for Distribution Estimation (MADE) [1] included an experiment that demonstrated that learning a handful of randomly generated orderings and connectivity masks as a form of regularization led to an increase the model's performance. However, the idea of multiple orderings was only touched upon. MADE's simultaneous simplicity and good statistical performance make it an ideal baseline and starting point for our experiments with FARM.

2.2 PixelCNN and Variants

2.2.1 PixelCNN

PixelCNN [2] represents one of the first attempts to use autoregressive models on images. In this approach, convolutional layers are used with causal masks. These causal masks are convolutions which have 0s in all entries to the right or below the current pixel, ensuring each pixel is only dependent on the rows above it and what is to the left of it. PixelCNN was shown to be more computationally tractable than other variants and started further research in this area.

2.2.2 PixelCNN++

PixelCNN++ [5] improved on the vanilla version by making changes in the distribution used to model pixel values among other changes. This bag of tricks led to state of the art performance at the time. For this work, we will use PixelCNN++.

2.3 Image Inpainting

Image inpainting is a well-known computer vision task that alters or fills in missing pixels of an image. The primary difficulty of image inpainting is the process of generating pixel values that are contextually similar to the existing nearby pixels; as a result, inpainting and completion tasks are often used to assess generative models as a qualitative manner for assessing how well they learned the underlying distribution. Both PixelCNN [2] and PixelRNN [6] used inpainting tasks to demonstrate both the effectiveness and diversity of their models.

2.4 Sublinear Sampling

This is the first work showing sublinear sampling with autoregressive models. All existing works [1, 2, 5] referenced in this paper and to our best knowledge, the literature as a whole, emphasize the linear nature of sampling.

3 FARM + MADE

We trained FARM variants of deep MADE architectures which learn several different orderings over the input pixels. In order to learn several orderings, we change the enumeration in the initial layer. This means the weight masks can be held constant and only the initial mask must be changed. We explore only simple variations of row raster orderings for now – alternate types of orderings, including diagonal and partially-random orderings, would be interesting future work. We treat these FARM+MADE models as ensembles over the several different orderings. Fig 1 shows two of the orderings we test in our experiments and how they can jointly be used to condition on more evidence when sampling a new pixel.

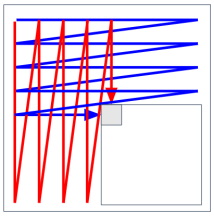

Figure 1: FARM + MADE with 2 orderings. This allows for conditioning on all available evidence.

We run experiments on a binarized version of the MNIST dataset [7]. As MNIST is fairly simple and well-studied, it serves as an ideal starting point to prove the effectiveness of the FARM approach on MADE before moving onto more complex models like PixelCNN++.

The model is trained with the standard MADE loss (i.e. negative log-likelihood or binary cross-entropy) modified to run over k orderings. Here the training data $\mathbf{x} = (x_1, \ldots, x_m)$ is an m-dimensional vector, θ represents the parameters of the MADE, and $\pi_1, \ldots, \pi_k$ are the k orderings.

$$\mathcal{L}_{\text{MADE}}(\theta; \pi_1, \ldots, \pi_k) = \\ \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\frac{1}{km} \sum_{i=1}^{k} \sum_{d=1}^{m} -x_d \log(p(x_d \mid \theta, \pi_i)) - (1 - x_d) \log(1 - p(x_d \mid \theta, \pi_i)) \right] \tag{1}$$

There are two ways this can be implemented in practice. The standard approach is Monte Carlo sampling where each minibatch $\mathbf{x}$ is fed through the network with a single specific ordering π_i. As we progress through the dataset in a single epoch, we periodically rotate out the ordering and

recompute the connectivity mask. Because we only recompute the connectivity mask once every several minibatches, the training process is a bit faster. This approach corresponds to training each model variant in the FARM ensemble separately, and is similar to the training procedure implemented in the original MADE paper.

On the other hand, at inference time we wish to leverage the ensemble nature of FARM with an averaging-over-experts approach. In particular, we wish for the k different model variants to output roughly similar joint distributions. So instead of running each minibatch through only one ordering, we instead introduce a method to train the model variants together by running each minibatch through every ordering. However, rather than naively averaging the output distributions and using the average for the binary cross-entropy loss, we instead perform a weighted average:

$$\hat{x}_d = \sum_{i=1}^{k} p(x_d \mid \theta, \pi_i) \cdot w_{i,d}, \tag{2}$$

where $w_{i,d}$ is a weight proportional to the amount of evidence (number of pixels) used by the model variant with the ith ordering in predicting the dth pixel, and $\sum_{i=1}^{k} w_{i,d} = 1$ for each d. Then we compute the loss as

$$\mathcal{L}_{\text{FARM+MADE}}(\theta; \pi_1, \ldots, \pi_k) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[\frac{1}{m} \sum_{d=1}^{m} -x_d \log(\hat{x}_d) - (1 - x_d) \log(1 - \hat{x}_d) \right]. \tag{3}$$

The reason that the weighting step is important is that some model variants will have much less available evidence than others in predicting a particular pixel, due to the different orderings. Computing the loss on the average would effectively "dumb down" the capability of the model variants with more evidence, because the quality of the average is limited by the expressivity of the model variants with little to no available evidence. Using the weighted average instead allows the loss to penalize mistakes in models with more available evidence more than mistakes in models with less available evidence and leads to significantly better sampling output than training with an unweighted average.

Note that another way to think about this second approach to the MADE loss as opposed to the first is that the Monte Carlo sampling is performed after swapping the two summations in Equation 1.

3.1 Weight Sharing

One of the advantages of using the deep MADE architecture for the multi-ordering ensemble is that it naturally involves weight-sharing across the models in the ensemble. Rather than having to learn k different sets of model parameters for k orderings, we only need to learn a single set of model parameters and train over each of the k orderings by modifying the connectivity mask on the very first layer of the MADE. This comes with a tradeoff. Because we are limited by the expressivity of the model, we can only train on a handful of orderings on a model with fixed size before the it begins underfitting.

3.2 Generalized Jensen-Shannon Divergence

We also explore the idea of using an additional loss term based on Kullback-Leibler divergence to encourage the different model variants in the ensemble to converge to the same distribution. The loss is mathematically similar to the generalized form of the Jensen-Shannon Divergence, which is:

$$\mathcal{L}_{\text{KL}} = \frac{1}{km} \sum_{d=1}^{m} \sum_{i=1}^{k} \mathcal{D}_{\text{KL}} \left(p(x_d|\theta, \pi_i) \middle|\middle| \frac{1}{k} \sum_{j=1}^{k} p(x_d|\theta, \pi_j) \right). \tag{4}$$

However, for the same reasons given previously, we will use a weighted version instead. This will appropriately penalize model variants based on how much evidence they have available to them, in order to prevent pushing high-evidence orderings and low-evidence orderings directly together and limiting expressivity. Here $\hat{x}_d$ and $w_{i,d}$ are the from Equation 2:

$$\mathcal{L}_{\text{KL}} = \frac{1}{m} \sum_{d=1}^{m} \sum_{i=1}^{k} w_{i,d} \cdot \mathcal{D}_{\text{KL}} \left(p(x_d|\theta, \pi_i) || \hat{x}_d \right). \tag{5}$$

Figure 2: An example of an occlusion test on 25 images. Left: original; middle: occlusions inpainted by MADE ($k = 1$); right: occlusions inpainted by FARM+MADE ($k = 8$).

3.3 Theoretical Analysis of System

For a truly optimal model ensemble, we should learn all $k = m!$ orderings. This is effectively set learning, where our ensemble can compute the probability for any single pixel conditioned on any subset of the other pixels. However, for virtually all practical purposes, $m!$ is far too high to explicitly learn that many orderings. Instead, we seek a middle ground. We demonstrate that learning only a handful of orderings is enough to observe tangible improvement in image inpainting tasks.

The FARM+MADE ensemble loss and the KL loss are both aimed at getting the model variants to converge their distributions, but they do so in two slightly different ways. The FARM+MADE loss encourages the weighted average of the distributions to approach the target, while the KL loss encourages the distributions to each approach the weighted average. Theoretically, employing both could help to improve the quality of sampled output.

3.4 Experiment 1: Image Inpainting

We report the results of testing variants of MADE on image inpainting tasks. There are three categories of inpainting tasks that are tested: the best case for FARM (i.e. occlusions are in the upper left), the average case (i.e. occlusions can be anywhere in the image), and the worst case (i.e. occlusions are in the bottom right). All occlusions are randomly rectangles. Every model is a MADE with two hidden layers, each with 1000 nodes. In each of the three cases, the models are evaluated on 100 randomly generated occlusions composed with 100 test set images each. We measure both the L_2 reconstruction loss per pixel of sampled output and the negative log-likelihood per pixel of the distributions provided by the models on the occluded regions only. We also investigated the various loss functions discussed in the theoretical section. Specifically, we tested FARM+MADE with the standard MADE loss, the FARM+MADE ensemble loss to train the model variants together, and the FARM+MADE loss with the KL loss to encourage similar distributions. The results are reported in Table 1. A visualization of the inpainting results can be seen in Figure 2.

	Best Case		Average Case		Worst Case	
	$L_2 \downarrow$	NLL $\downarrow$	$L_2 \downarrow$	NLL $\downarrow$	$L_2 \downarrow$	NLL $\downarrow$
MADE ($k = 1$) baseline	0.0920	0.2611	0.1493	0.3461	0.0832	0.2373
FARM + MADE ($k = 2$)	0.0657	0.1609	0.1339	0.2849	0.0835	0.2139
FARM + MADE ($k = 4$)	0.0642	0.1433	**0.1325**	0.2617	**0.0805**	0.1843
FARM + MADE ($k = 8$)	**0.0641**	**0.1390**	0.1340	**0.2589**	0.0810	**0.1798**
FARM + MADE ($k = 2$) + FARM loss	0.0684	0.1591	0.1397	0.2958	0.0865	0.2313
FARM + MADE ($k = 4$) + FARM loss	0.0702	0.1570	0.1466	0.3024	0.0886	0.2153
FARM + MADE ($k = 8$) + FARM loss	0.0733	0.1854	0.1578	0.3506	0.0963	0.2515
FARM + MADE ($k = 2$) + FARM loss + KL	0.0682	0.1572	0.1402	0.2930	0.0865	0.2281
FARM + MADE ($k = 4$) + FARM loss + KL	0.0687	0.1597	0.1456	0.2991	0.0898	0.2107
FARM + MADE ($k = 8$) + FARM loss + KL	0.0734	0.1761	0.1567	0.3394	0.0966	0.2482

Table 1: Results for image inpainting. Arrows indicate objective.
We see FARM outperforms the baseline in every case.

We see that a family of 8 orderings significantly outperforms a single-ordering MADE. Interestingly, the addition of the FARM loss and the KL loss make the ensemble slightly worse than the standard MADE loss, although they still improve upon the baseline single-ordering MADE. We theorize that this could be linked to the weak expressivity of MADE as a small, two-layer network, because the

extra losses create a burden to match distributions that can only be optimized by sacrificing the quality of the output. The same ideas could prove useful for a more complex model, such as PixelCNN++.

3.5 Experiment 2: Faster Sampling

Normally, autoregressive models generate sequentially, meaning in this case each pixel is generated conditioned on the pixels that were generated prior. However, using multiple orderings and the conditional independence assumptions that each comes with, we can fill in several pixels during one step; with eight orderings, for example, eight pixels can be filled in at once. This method can theoretically lead to sub-linear time generation of a new image, with the optimal orderings used dependent on the dimensions of the image.

Note that generation with multiple orderings will inevitably reduce performance; indeed, the independence assumptions used for generation from each ordering will fail, as inferences are made using orderings on data that was not present the step before. However, if the hypothesis spaces created from learning these orderings are sufficiently overlapping (i.e. have low KL divergence), then the quality of the generated images will not be as bad, since the predictions for each of the orderings will be roughly similar.

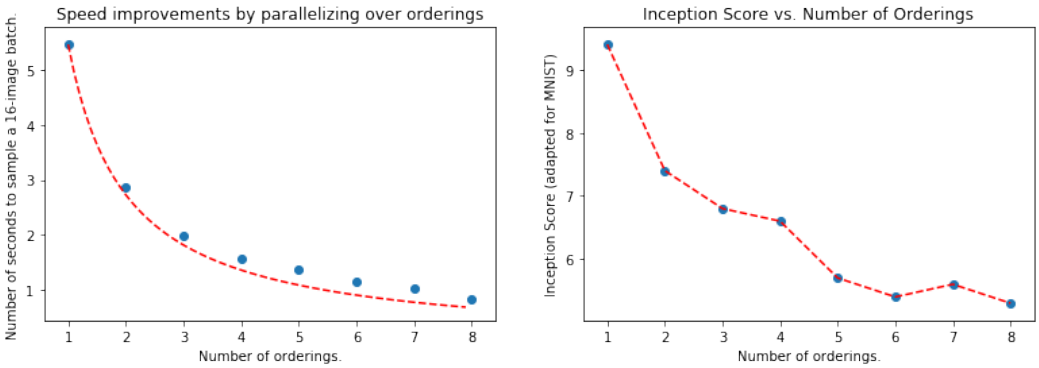

Table 2: Visualizations of image sampling.

Table 2 shows that generation using FARM will significantly decrease sampling time when multiple orderings, particularly variants on raster scans, are used for sampling. These speed improvements occur close to the optimal sub-linear amount; however, optimal sub-linearity is not fully achieved. This is likely due to suboptimal selection of the orderings to generate from, as a truly efficient ordering would take advantage of the diagonal, which these orderings do not. Nonetheless, these sub-linear time improvements emphasize the degree to which multiple orderings can improve sampling time needed.

Likewise, Table 2 also demonstrates the decrease in generated image quality that sampling from multiple orderings causes - as the number of orderings used to sample from increases, the lower the Inception Score (using Inception trained on MNIST rather than CIFAR). In particular, there are large drops between using 1 and 2 orderings and using 4 and 5 orderings. We hypothesize that this occurs due to an increasing number of "collisions" that occur between the different orderings being sampled from, as well as the similar structures that occur between different digits in MNIST (e.g. the stem of a 7 is similar to the stem of a 9).

However, our results show that despite lower sample quality using multiple orderings, we can attain a sampling time that is not necessarily $O(n)$. Indeed, assuming conditional independence between pixels that are far away from each other allows us to attain sub-linear sampling times that are close to the theoretical optimum. Moreover, better performance could be attained by training FARM+MADE using a KL-style loss function for longer amounts of time, which would bring each ordering's hypothesis space closer to one another, or by using more optimal orderings (i.e. that could take advantage of the diagonal).

4 FARM + PixelCNN++

We now look at adapting FARM to PixelCNN++, a more recent method which has variants that are still actively used. We repeat a similar process as with MADE using this model except now on the CIFAR dataset as PixelCNN related results are reported on this benchmark as opposed to MNIST due to its greater expressivity.

In our formulation, we train two different versions with the causal masks as in Fig. 3. Due to the long and expensive training procedure, we were unable to explore more masks.

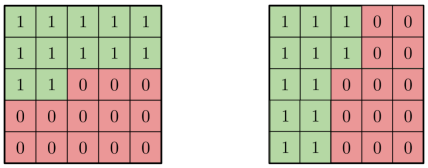

Figure 3: PixelCNN++ causal masks. The left is the canonical version.

4.1 Weight Sharing

Weight sharing is very important in FARM to provide a computationally inexpensive training procedure. For PixelCNN++, we propose learning a standard convolutional kernel with the exception that the center value is 0. During train time, for each batch a causal mask is passed as a parameter and the network is accordingly masked. In this setup, the two trained masks from above would share the same weights in the top left region.

Unfortunately, as training each variant of PixelCNN++ takes several days on a P100, we were unable to afford experimenting with this after training the baseline and running an experiment with JSD loss. Additionally, the implementation for this becomes quite tricky, as PixelCNN and PixelCNN++ actually use two convolutional kernels to effectively simulate the one on the top. This is used to remove a blind spot in the receptive field as detailed in [2]. The implementation also doesn't mask the kernels, but rather does padding and shifting. These changes mean weight sharing demands a near complete rewrite of the model.

4.2 Jensen-Shannon Divergence

We again attempted to use a Jensen-Shannon Divergence loss in order to push the two causal mask models together to a similar distribution. As a result, the new loss function was:

$$\mathcal{L}_{\text{JSD}}(\theta; \pi_1, \pi_2) = \mathcal{L}_{\text{PixelCNN++}}(\theta; \pi_1, \pi_2) + \text{JSD}(p(x \mid \theta, \pi_1) \parallel p(x \mid \theta, \pi_2)) \tag{6}$$

where $p_\theta(x|\pi_1)$ and $p_\theta(x|\pi_2)$ are the output distributions from each causal mask on input x. Note that the latter term is equivalent to the unweighted loss in Equation 5.

In order to measure the impact of this term, we first train two versions with the masks from Fig 3 and achieve losses of 3.05, 3.05 respectively on the test set. Note that this is slightly higher than what is reported in the paper for the first mask as we were unable to fully train it to convergence due to budgetary restrictions. When training with the loss term, we see losses of 3.31 and 3.30 respectively on the test set using the standard loss without the JSD term. When looking into why we have worse results, part of it might be because the JSD term dominated too much, and forcing the orderings together restricted their expressivity. With more resources, we still might get better results if we massage this parameter.

4.3 Theoretical Analysis of System

The weight sharing scheme described in this section is distinctly different from in FARM + MADE. While the previous case showed learning all orderings isn't possible, in this case learning all orderings is feasible. If we follow the training procedure described, we are essentially learning a standard convolutional filter with the exception that the center pixel must be 0. The restriction to causal masks

might serve as a form of regularization which is akin to dropout in a convolutional setting. In fact, the original MADE paper [1] references how attempting to learn multiple orderings might serve as a form of regularization. This might also be a particularly useful given that PixelCNN++ [5] uses regularization and shows without it the model overfits.

Once again, we see that the JSD term we used in the loss function is unweighted, so it is problematic in cases where one of the masks has a lot of evidence but the other has little, This scenario forces the "smart" mask to perform worse as the "dumb" mask is unable to give better results given a lack of evidence. This, in conjunction with poorly tuned parameters, might be the restricting factor in better results, and is something we would like to explore given more funding. In particular, again we think weighting the parts of the JSD based on how much evidence they have is a good approach. While we were able to realize this and rerun experiments for MADE, due to training time and cost limitations, this was not possible for PixelCNN++.

4.4 Experiment 1: Image Inpainting

We now apply our system to the task of image inpainting. In this setup, we take a batch of 16 images from the CIFAR dataset and 0 out part of the image. In order to fill this image, we sample using the standard canonical mask as a baseline, and use FARM as our method where we use the mask with the most amount of evidence or a weighted mean of both masks when possible. For example, if FARM decides the canonical mask is better, in the first pixel of every row since the rotated mask can be used, the distributions of both are combined to sample that pixel.

We split this into two challenges: best case and average case. Best case is when the occlusion is in the top right, so the canonical mask only has minimal information from the top row of pixels but the rotated mask has information from the entire side. The average case is a random occlusion in the middle of the image. Note that in this case, we don't do a worst case category as then FARM and the baseline use the canonical mask. As we don't do weight sharing, we expect the exact same results as opposed to MADE where weight sharing might mess with the canonical approach. For each case, we run 15 batches to get a total of 240 images.

	Best Case		Average Case	
	Bits/Dim ↓	Inception Score ↑	Bits/Dim ↓	Inception Score ↑
PixelCNN++	15.53	6.10	11.61	7.21
FARM + PixelCNN++	**14.95**	**6.62**	**11.50**	**7.24**

Table 3: Results for image inpainting. Arrows indicate objective. We see FARM outperforms the baseline in all cases.

The results in Table 3 show FARM clearly outperforms the baseline. While the margin is more significant in the best case, it still holds true for the average case. Note that we only primarily care about bits per dimension as log likelihood is tractable, but inception score is provided for completeness. Note that bits per dimension is simply scaled log likelihood and used to be consistent with the literature [2] [5].

We also provide a few cherry picked visualizations showing the improvement in Table 4. In general, we see that the traditional ordering often fails when it is unable to accurately understand the local texture in the occlusion, which FARM is able to better detect due to more information. This can be seen in the top two rows and the left image in the bottom row. By contrast, the few cases where the traditional method beats FARM, which means FARM chooses the sideways ordering even though the canonical ordering performs better, is generally when the occlusion blocks something which is more semantic, making the image inpainting challenge harder. This can be seen in the images of the moose in the bottom right.

4.5 Experiment 2: Faster Sampling

Traditional sampling methods fill in images with autoregressive models by filling in the next available pixel. However, due to conditional independence assumptions made in the orderings, we can often fill in more than one pixel. This becomes even more prominent when we have two orderings, where we can use each ordering to fill in a few pixels at once. In particular, we can perform very fast sampling by sampling the entire diagonal at once at each step and obtaining $O(\sqrt{n})$ sampling.

Table 4: Visualizations of image completion.

The proposed sampling procedure using two orderings actually makes a few more additional independence assumptions than just those by local connectivity as it limits the receptive field slightly. However, PixelCNN++ [5] found limiting the receptive field did not have a huge impact on performance, so we conclude this assumption is safe to make. It is important to keep in mind the level of this assumption can be changed to produce a tradeoff between speed and quality. Due to computational limitations in sampling a sufficient amount of images to evaluate quality, we only take two extremes, but with more resource one could identify the curve and pick an optimal point based on needs.

	Inference Time ↓	Inception Score ↑
PixelCNN++	192.57 ± .75s	3.95
FARM + PixelCNN++	28.92 ± .18s	3.79

Table 5: Results for faster inference. Arrows indicate objective. We see clear tradeoffs.

Table 5 shows faster FARM sampling results in a massive performance gain, cutting sampling time for about 3.5 minutes to half a minute. In addition, while there is a drop in Inception Score as expected due to limited receptive fields, the drop is surprisingly small.

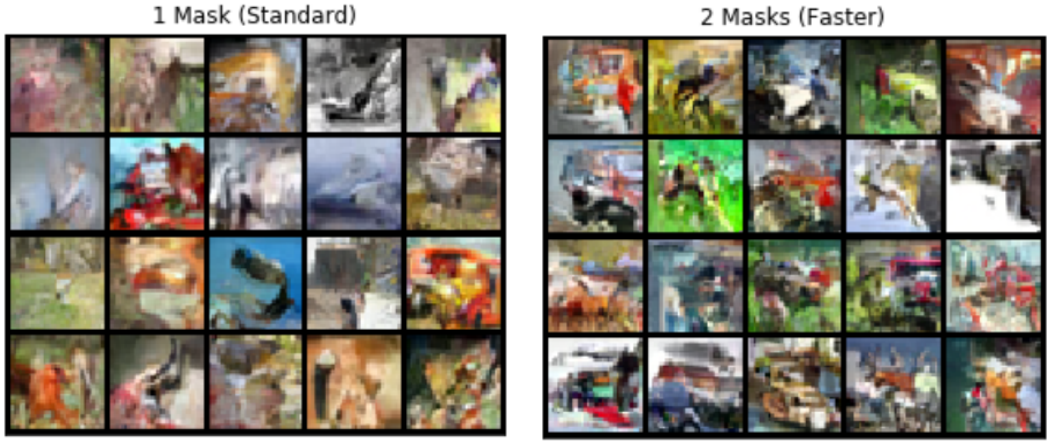

Table 6: Visualizations of image sampling.

When looking at visualizations, which were randomly selected, we see far worse and inconsistent texture in the faster generated samples. As such, we hypothesize the Inception Score does not drop a huge amount because it is already so low and the classifier can barely tell what anything is.

While the results show this fast sampling is worse, the key finding is that sampling autoregressive models in images is not strictly $O(n)$. By assuming certain conditional independence, which is very reasonable given that pixels on opposite sides are likely lowly correlated, we can get sub-linear sampling, an astounding property that isn't otherwise generally true or feasible for domains like text. Due to computational limitations, we were not able to generate a full tradeoff curve, but there might be a better point that is considerably faster with little degradation. Also, we provide one sampling strategy is sublinear, but it is easy to see that there are many, and there could be several that outperform the provided one.

5 Conclusion

In this work, we sought to learn FARM, a **F**amily of **A**uto**R**egressive **M**odels, as a means to overcome limitations in generative autoregressive models: dependence on a fixed ordering and $O(n)$ sampling times. We first detailed methods of effectively learning FARM and the ways in which it could **improve** on existing models' tractability and ensemble cohesiveness, including weight sharing and new loss functions. We also provided some theoretical insight into these structures, such as discussing how MADE weight sharing is akin to set learning but PixelCNN++ weight sharing is more like regularization. These insights highlighted some of the **tradeoffs** between train time, coheseviness, performance, and speed.

We then performed experiments training families of MADE and PixelCNN++ trained on MNIST and CIFAR-10 respectively, assessing both on image inpainting tasks as well as sampling time and quality versus the number of orderings. In particular, we demonstrated that using FARM with MADE or PixelCNN++ gave **(1)** better results for image inpainting over the standalone model and **(2)** provided the first method of sub-linear time sampling for autoregressive models with some performance tradeoff.

6 Future Work

In this work, we provide an interesting exploration into a relatively untouched area of autoregressive models. As much of this work is driven by intuition which we seek to verify through experiments, there are many tests we proposed and ran to better understand how FARM might work. Unfortunately, one limitation to this work was the level of funding. Along with the base $150 given to the three group members, we requested an additional $100 and were able to scrounge up another $200 from other sources. This was quickly exhausted due to the long and expensive training time for PixelCNN++ and the slow time to generate sufficient images for evaluation of sample quality.

With more funding, there are many more experiments we would like to run. The biggest are in PixelCNN++, which was extremely difficult to play around with due to long training times. We would like to firstly implement the weight sharing idea, which we think would really improve the computational feasability of the system. We would also like to try the weighted JSD term along with some hyperparameter tuning for weighting this term.

When it comes to inpainting, we would like to combine the faster sampling and image inpainting experiments. We believe this would result in a smaller performance degradation than sampling the entire image, as in the latter there needs to be much more care in ensuring the various masks are consistent to each other based on less evidence. Unfortunately, the cost of sampling enough times was too high to run it for this work.

More broadly, we believe it would be interesting to explore this approach in PixelSNAIL. This variant requires even more resources to train, but it uses attention to capture more global structure, which could be helpful in keeping the masks consistent. It could also help FARM outperform the baseline as increasing the amount of evidence with FARM generally results in more evidence which is far away, which is more impactful in this variant.

Lastly, we think using FARM with a modified VQ-VAE-2 might be useful for image inpainting that challenges state of the art methods. This requires some reformulation in getting an encoder that preserves which region is occluded in the latent space, but seems to be promising.

References

[1] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. *CoRR*, abs/1502.03509, 2015.

[2] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016.

[3] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, Norman Casagrande, Dominik Grewe, Seb Noury, Sander Dieleman, Erich Elsen, Nal Kalchbrenner, Heiga Zen, Alex Graves, Helen King, Tom Walters, Dan Belov, and Demis Hassabis. Parallel WaveNet: Fast high-fidelity speech synthesis. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3918–3926, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[4] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *CoRR*, abs/1906.00446, 2019.

[5] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017.

[6] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016.

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