LABORATORY DIRECTED RESEARCH & DEVELOPMENT
WHERE INNOVATION BEGINS

SAND21XX-XXXXR
LDRD PROJECT NUMBER: 224121
LDRD PROJECT TITLE: Mapping Stochastic Devices to Probabilistic Algorithms
PROJECT TEAM MEMBERS: Brad Aimone, Sasha Safonov

ABSTRACT: Probabilistic and Bayesian neural networks have long been proposed as a method to incorporate uncertainty about the world (both in training data and operation) into artificial intelligence applications. One approach to making a neural network probabilistic is to leverage a Monte Carlo sampling approach that samples a trained network while incorporating noise. Such sampling approaches for neural networks have not been extensively studied due to the prohibitive requirement of many computationally expensive samples.

While the development of future microelectronics platforms that make this sampling more efficient is an attractive option, it has not been immediately clear *how* to sample a neural network and *what* the quality of random number generation should be. This research aimed to start addressing these two fundamental questions by examining basic "off the shelf" neural networks can be sampled through a few different mechanisms (including synapse "dropout" and neuron "dropout") and examine how these sampling approaches can be evaluated both in terms of evaluating algorithm effectiveness and the required quality of random numbers.

INTRODUCTION AND EXECUTIVE SUMMARY OF RESULTS:

Probabilistic neural algorithms offer considerable potential for critical computing tasks such as forecasting, prediction, and Bayesian inference and these algorithms represent a significant advance beyond today's deterministic artificial neural networks. Algorithms that leverage Monte Carlo sampling approach maximal impact with an ever-increasing number of samples. Often, the requirement of many samples makes their use prohibitive – while conventional hardware, like CPUs and GPUs can implement random sampling, neural network algorithms are not typically optimized for sampling. Developing novel computing architectures that leverage devices with the required stochasticity would provide a powerful means to implement these algorithms at unprecedented scales.

While some emerging architectures, such as neuromorphic, can be proficient in at-scale Markov chain sampling, the limitation of the conventional random number generators in these systems limits their ability to attain the precision and flexibility necessary for critical results. The algorithmic need for stochastic fidelity presents a unique microelectronics co-design challenge:

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Sandia National Laboratories

U.S. DEPARTMENT OF ENERGY

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

how do the specific properties of devices, such as their mechanism and deployment of stochasticity, impact the behavior of algorithms whose designs assume the presence of an idealized source of randomness? Stated differently, how do we assess how different forms of random number generation impact probabilistic algorithm performance?

This research aimed to start developing a formal approach to benchmark probabilistic algorithm performance while changing the constituent distributions of the random numbers provided by emerging devices. Here we considered two key research questions related to probabilistic neural algorithms: (1) does probabilistic sampling contribute an added capability to neural networks?, and (2) how does the precision of random numbers impact neural network.

DETAILED DESCRIPTION OF RESEARCH AND DEVELOPMENT AND METHODOLOGY:

We performed two experiments. Experiment 1 focused on investigating the effect of reduced precision in a novel stochastic neural network paradigm, and Experiment 2 focused on exploring existing methods for stochastic sampling of neural networks. Experiment 1 is explained in detail in the Addendum.

Experiment 1

Network: *Training* - We defined here a new class of artificial neural network that we refer to as scANNs (sampling by coinflips ANNs). These networks are trained conventionally with only minor additional constraints (e.g., weights restricted between '0' and '1'). *Inference* – During inference, we convert the ANNs weights to a Bernoulli probability of being a '1' or '0'. For instance, if a weight equals 0.7, for a given trial the weight has a 70% of being '1' and a 30% chance of being '0'. For the experiments we explored, all weights in the network are treated as probabilistic.

Experiment: We trained simple 3-layer ANNs (784-400-10) on either Fashion MNIST [8] or MNIST [5] data. We then convert to sampling networks, where we sample the networks 1000 distinct times and quantify the outcomes of the test data. Each 'sample' effectively receives one vote, and the samples are combined to observe the distribution of network votes.

Reduced Precision: We then imposed reduced precision networks by converting the weights of the scANN networks to 8-bit and 6-bit precision. Thus for instance, a weight = 0.51 would be converted to 0.5078125 (the nearest 8-bit representation) and 0.50 (the nearest 6-bit representation).

Experiment 2

Network: We fine-tuned the Torchvision VGG16 network [7] that was pretrained on ImageNet [3], on ImageNet with the new dropout method. For training, stochastic gradient descent with a learning rate of 0.01 and a momentum of 0.9 was used. The model was fine-tuned for 30 epochs, with the best performing epoch being used (best accuracy on the validation set for ImageNet).

Experiment: Monte Carlo dropout is a variational inference method that can be applied to any network that is trained with dropout [1]. This method involves multiple forward passes with dropout during inference, giving us mean and standard deviation values. This can be used to help combat over-confident predictions in out-of-distribution inputs. 10 forward steps were used for mc_dropout evaluation.

RESULTS AND DISCUSSION:

Experiment 1

Full results from Experiment 1 are included in the Addendum. Briefly, our results demonstrated that

- The scANN approach to sampling a normally trained network does not significantly impact performance
- The scANN approach provides an observable distribution of uncertainty in which more confusing (e.g., wrong) test data points exhibit less confidence in their sampling ensemble classification.
- The precision of random number generation does noticeably affect performance at 6-bit, but not significantly at 8-bit (Figure 1). Further, it appears that more sampling can overcome some of this penalty for lower precision.

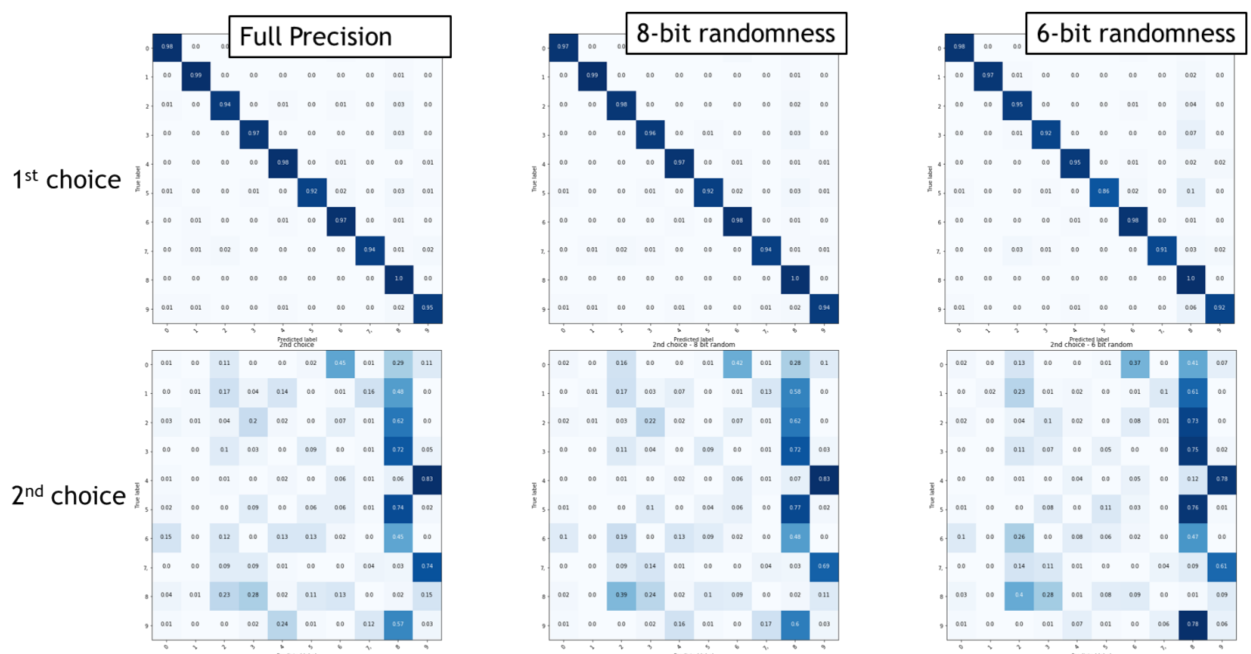


Figure 1: Precision of random numbers only modestly impacts observed distributions of scANN classifications

Experiment 2

We compared three different categories of Monte Carlo dropout networks to examine the relative performance of these networks. We specifically looked at the adversarial ImageNet dataset which consists of highly challenging examples. For measuring how well uncertainty is calibrated, we employ uncertainty calibration error (2).

Monte Carlo Dropout

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} \sigma_{SM}(f_{w_i}(x))$$

Uncertainty calibration error

$$UCE := \sum_{m=1}^{M} \frac{|B_m|}{n} |err(B_m) - uncert(B_m)|$$

$$err(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \mathbf{1}(\mathrm{y}'_i \neq y)$$

$$uncert(B_m) = \frac{1}{|B_m|}\sum_{i \in B_m} \boldsymbol{H}(\boldsymbol{p}_i)$$

Where *Bm* is the number of confidence predictions that fall into a bin, where H() is the entropy measurement

To briefly summarize the results reported in Table 1.

- While accuracy and other performance related metrics were similarly low across all variations of the model tested (adversarial ImageNet is a particularly challenging dataset), in general Monte Carlo dropout reduced UCE.
- Additional fine tuning with the new dropout paradigm further reduced UCE and gave the best calibration performance.

Table 1: Synapse Monte Carlo dropout appears effective at reducing uncertainty calibration error

DROPOUT	FINE-TUNING	ACCURACY	UCE
None	N/A	**.0272**	.6243
Neuron	No	.0252	.5806
Neuron	Yes	.0227	.5352
Weights	No	.0241	.6150
Weights	Yes	.0248	**.5196**

<u>*Discussion of Results*</u>

Combined, Experiments 1 & 2 provide concrete, albeit preliminary, evidence that Monte Carlo sampling techniques for neural networks can be effective at characterizing the inherent uncertainty contained within training data sets. In both experiments, the networks investigated were trained in a standard, deterministic manner. This is in contrast to some previous methods that leverage synaptic sampling that often incorporate stochasticity into the training process as well [1, 6]. It is possible, and perhaps preferable, that the stochastic sampling approaches that we examine are similarly incorporated into the training phase of the model, but the ability to operate on fully trained models has advantages as well.

These experiments also highlight the high computational costs of Monte Carlo sampling of neural networks. The simple networks in Experiment 1 needed to be sampled a thousand times to provide estimates of network uncertainty, and even this did not appear to fully saturate performance. This slow convergence of approximating a distribution is a known challenge with Monte Carlo, and it is inherently associated with high computational costs. In Experiment 2, the high costs of Monte Carlo sampling of large VGG networks limited the number of samples to less than 100 samples, which was also a limitation in prior published efforts.

Finally, Experiment 1 demonstrated that the sampling approach was robust to Bernoulli probabilities of rather limited precision (down to 6-bits of precision, although lower is possibly sufficient as well). This is important because as hardware-level solutions for implementing ANN sampling are explored, it may not be reasonable to assume high precision random numbers will be widely available.

ANTICIPATED OUTCOMES AND IMPACTS:

The two key results of the Experiments described above – that stochastic sampling can be an effective, albeit expensive, approach to describing the uncertainty of neural networks and the fact that this Monte Carlo sampling can tolerate lower precision random numbers – are important observations in the longer-term goal of exploring new hardware and microelectronics approaches to further probabilistic computing technologies.

The development of a new microelectronics solution for stochastic computation, which will be a target in the upcoming DOE Energy-Efficient Microelectronics program, will require a number of advances across many technology scales. While stochastic and Bayesian algorithms are believed to be an extremely important direction for artificial intelligence and scientific computing, their utility is limited considerably by the high computational costs of repeated sampling and random number generation. While specialized architectures such as GPUs can accelerate deterministic linear algebra computation; they are not necessarily good for repeated stochastic operations. Conventional random number generation confers a tremendous cost in conventional architectures, so one possible path to achieving more efficient Monte Carlo sampling is to generate random numbers at the device level.

Moving towards this vision of a novel approach to computing that makes random numbers inexpensive through ubiquitous device-level stochasticity will be a complicated task. Part of the challenge in generating an entirely new technology stack for probabilistic computing is how do we evaluate how characteristics at the device level will impact algorithm performance, and vice versa (how requirements at the algorithm level will translate to device needs). The research here is a critical first step in identifying an initial set of metrics by which the precision of random numbers, which presumably will be a device level consideration, may impact algorithm performance.

Future work in this direction will focus on optimizing these algorithm approaches and exploring the impact of stochastic sampling of synapses on more complex data sets. If successful, this research will help lead to a much more extensive exploration of probabilistic neural networks. Further, the results from this project can immediately be used to help guide microelectronics research at a number of other scales, particularly at the device and circuits level.

CONCLUSION:

The development of a novel probabilistic computing system that takes advantage of stochasticity of novel microelectronics devices introduces significant challenges across technology scales. The research explored in this project helps understand the potential for stochastic sampling of synaptic weights within artificial neural networks, which is a potential significant application of a probabilistic computing system.

In summary, this project illustrated that Monte Carlo sampling of synaptic weights in neural networks can be useful for characterizing the uncertainty of neural network classifications, and further highlights that these approaches are likely tolerant to reduced precision of the key random number generation component of the stochastic sampling. Identifying that the mechanisms of stochastic sampling explored here are tolerant of lower precision is an important step in understanding the requirements of a future probabilistic computing system.

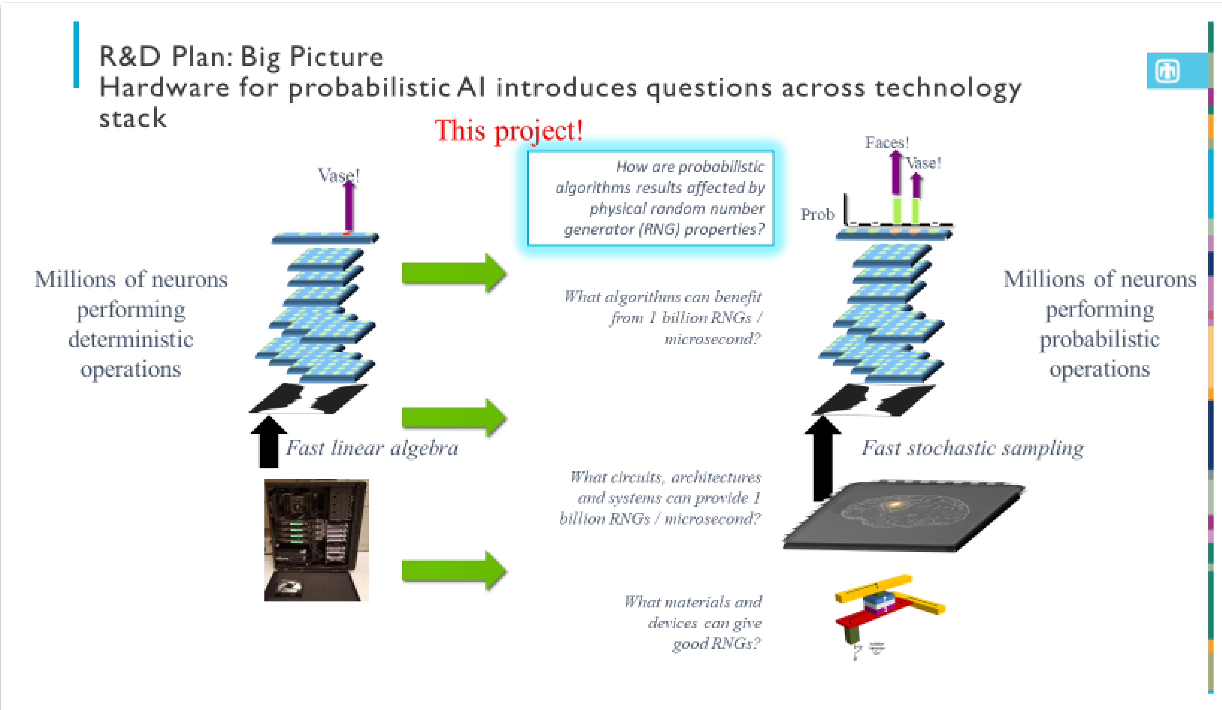
REFERENCES:

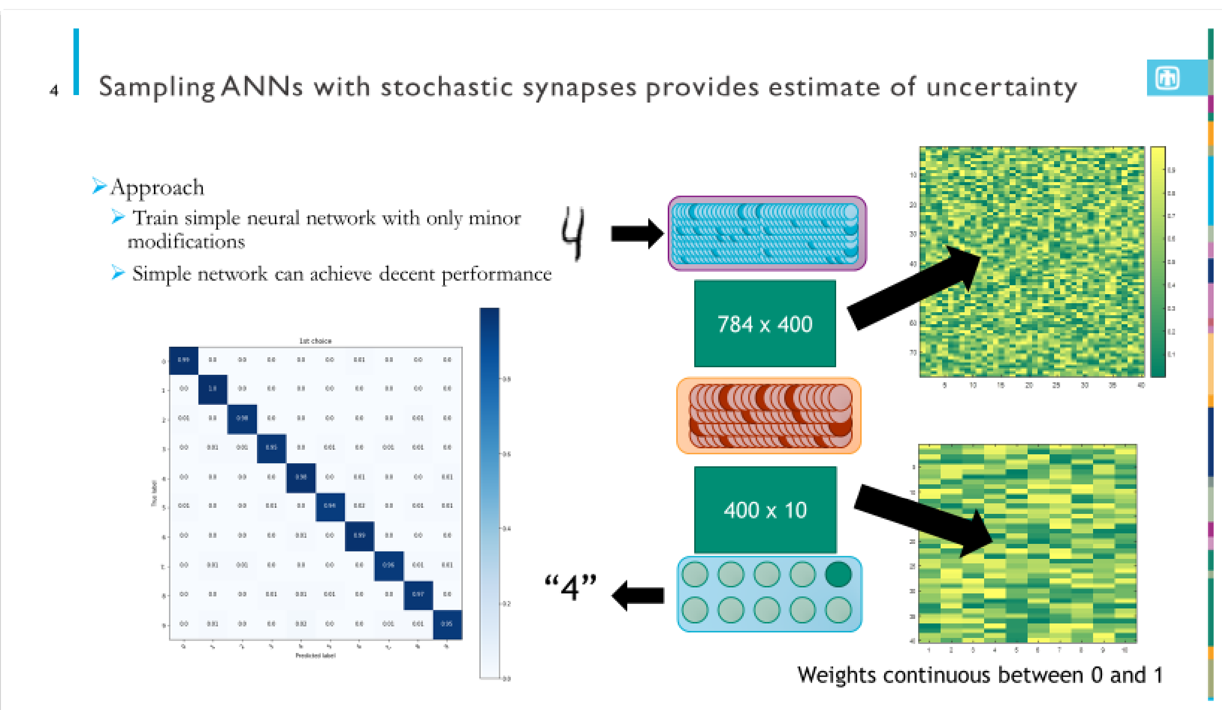
1. Al-Shedivat, Maruan, et al. "Inherently stochastic spiking neurons for probabilistic neural computation." *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE, 2015.
2. Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." *international conference on machine learning*. PMLR, 2016.
3. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.
4. Laves, Max-Heinrich, et al. "Well-calibrated model uncertainty with temperature scaling for dropout variational inference." *arXiv preprint arXiv:1909.13550* (2019).
5. LeCun, Yann. "The MNIST database of handwritten digits." *http://yann. lecun. com/exdb/mnist/* (1998).
6. Neftci, Emre O., et al. "Stochastic synapses enable efficient brain-inspired learning machines." *Frontiers in neuroscience* 10 (2016): 241.
7. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
8. Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747* (2017).

ADDENDUM:

We include two Addendums, the year-end presentation to the Investment Area and a brief report summarizing the methods and results from Experiment 1 in more detail.

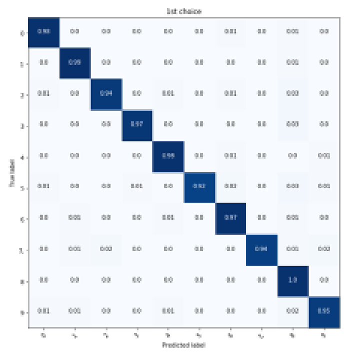
Addendum 1

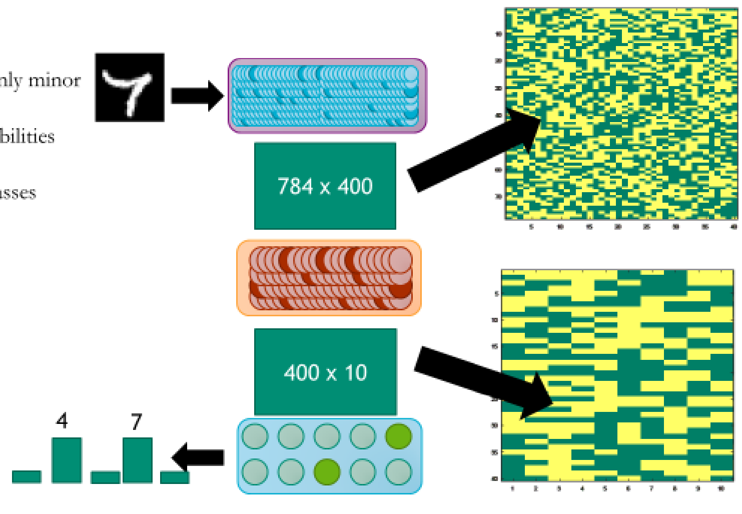

R&D Plan: Big Picture
Hardware for probabilistic AI introduces questions across technology stack
This project!
Vase!
Millions of neurons performing deterministic operations
Fast linear algebra
How are probabilistic algorithms results affected by physical random number generator (RNG) properties?
What algorithms can benefit from 1 billion RNGs / microsecond?
What circuits, architectures and systems can provide 1 billion RNGs / microsecond?
What materials and devices can give good RNGs?
Faces!
Vase!
Prob
Millions of neurons performing probabilistic operations
Fast stochastic sampling


4
Sampling ANNs with stochastic synapses provides estimate of uncertainty
Approach
Train simple neural network with only minor modifications
Simple network can achieve decent performance
4
784 x 400
400 x 10
"4"
Weights continuous between 0 and 1

Sampling ANNs with stochastic synapses provides estimate of uncertainty

- Approach
  - Train simple neural network with only minor modifications
  - Convert weights to Bernoulli probabilities (weighted coinflips)
  - Sample network to identify what classes

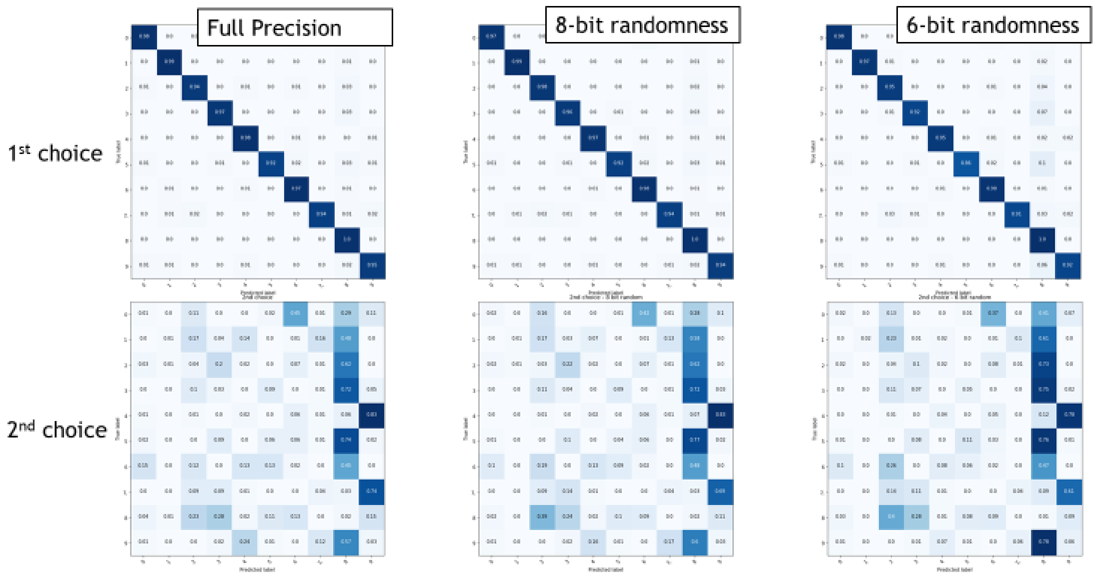




Weights *sampled* as probability

2nd choice of stochastic sampled networks is often the 'right' answer

7 Sampling ANNs with stochastic synapses is robust to low precision synapses



8 Is synaptic sampling related to neural network *dropout?*

- Dropout is an important method for neural network training
  - Neurons are randomly bypassed / silenced during training epochs
  - Ensures that large networks do not overfit on data
- Dropout can be used *during inference* as an estimate for uncertainty
  - Different method than stochastic synapse sampling above
  - Preliminary data: examine performance on natural adversarial dataset (really hard examples)
  - Measuring "uncertainty calibration error"
- One conclusion – these MC approaches are *really* expensive (typically people only use 50-100 samples)

"lighthouse" "umbrella" "broom"

Dropout	Fine-Tuning	Accuracy	UCE
None	N/A	.0272	.6243
Neuron	No	.0252	.5806
Neuron	Yes	.0227	.5352
Weights	No	.0241	.6150
Weights	Yes	.0248	.5196

9

PI's PROJECT LEGACY

- Preliminary evidence that sampling neural networks is promising
  - Does not significantly impair performance
  - Uncertainty introduced by sampling appears to match intrinsic uncertainty in data
- Sampling neural networks has high implementation cost on conventional systems
  - Highlights the value of a computational system designed for more efficient Monte Carlo sampling
- Stochastic Synapse Sampling approach appears robust in presence of low-precision sampling
  - Any hardware with device level randomness will likely be limited to a low stochastic precision

10

PROJECT OUTPUTS, CAPABILITIES, TEAM BUILDING

Project Outputs

- Intellectual Property (TAs, patents, copyrights)
  - TA #15896 filed on **scANNs: Sampling through Coinflips Artificial Neural Networks**
- Publications (in preparation or review)
  - Conference paper in preparation on **Synaptic Sampling of Neural Networks**

Capabilities

- Metrics to look at stochastic neural networks and impacts of precision

Team Building and Partnerships

- Results are highly relevant to provide to new COINFLIPS community

Addendum 2: Pre-Submission Draft of Sampling COINFLIPS Artificial Neural Networks paper

DRAFT – Synaptic Sampling of Neural Networks– DRAFT

Brad Aimone

September 16, 2021

1 Overview

Question: Does it make sense to synaptically sample trained neural networks?

It is well known that sampling neural networks during training (such as with Dropout) is very effective at providing a robust training that mitigates the impact of overfitting. Dropout is an example of neural sampling; whereby neurons are randomly removed in training, pushing learning in each epoch to a different subset of neurons. There is also a process of synapse dropout, which is implicit in neuron dropout (all neurons removed by definition have their synapses removed for those training epochs), but would be more finely administered.

What is less immediately obvious is whether there is a value of sampling neural networks during inference? In the case of inference, the value is not one of regularization, but rather one of assessing the uncertainty inherent in a network through a sampling process. Because of its scale, neuron dropout during inference is pretty disruptive; however it is possible that a process similar to synapse dropout may be more suitable for inference mode. Here, we examine the value of synapse sampling in neural networks.

At first glance, it is not immediately apparent that the value of synaptic sampling a trained neural network outweighs the costs. If a network can be trained to provide in one-pass a good answer or prediction, why would you want to sample that network hundreds, or even thousands, of times? Each sample will be less accurate than the determinsitic network, and it is not obvious that in aggregate the accuracy will be higher. Further, the computational cost of sampling is high, with inference costs already being a complicating factor for neural networks.

As will be described below, the process of synaptic sampling we propose has several advantages that help mitigate some of the concerns. Specifically, the method we propose allows us to

2 Approach to synapse sampling neural networks

While ANNs can become quite complex, we consider that each layer of a neural network effectively has the following form

$$x_B = f(x_A * W_{AB} + b_B) \tag{1}$$

where x_A is the activation of neurons in layer A, x_B is the activation of neurons in layer B, W_{AB} is the weight matrix, aka synaptic strengths, between A and B, b_B is a bias term on each of nodes, and $f()$ is the activation function of neurons, such as a relu or sigmoid. We consider a specific element of W_{AB} as w_{ab}, which is the weight of the connection between node a of A to node b of B.

The vector matrix multiply within the function is, of course, a linear mapping, and while there are some constraints on f, in general the main requirement for ANNs is that it be non-linear and (mostly) differentiable. For most neural networks, the individual element weights of W can be any number, positive or negative, though restricted precision networks begin to introduce constraints in both range and precision of the weights.

The approach we take here for synapse sampling is to consider W_{AB} as a matrix of *probabilities* instead of continuous valued weights. This requires three modifications of a standard neural network process

1. Either restrict weights in training to be between 0 and 1, or renormalize network (through modifying b and f) to allow weights to be between 0 and 1. This should not have a dramatic impact on performance for standard activation functions if weights are allowed arbitrary precision

2. After training, the weight matrix W_{AB} is thus viewed as a probability matrix P_{AB}, whereby the probability of each element being equal to 1 is w_{ab}. For each inference sample, instead of using W, we will instead

use a sample matrix S_{AB}, where each element s_{ab} is equal to either 1 or 0, with stochastic draw through a *weighted coinflip* with probability w_{ab} of equaling 1.

3. During inference, we then run many sampling inference cycles, each with a **different** S_{AB}. Each sample will provide a set of predictions based only on using those weights that were included in that particular sample.

The expected value of the vector matrix multiply $x_A * S_{AB}$ is the same as $x_A * W_{AB}$, however each individual sample of the former will be different.

Obviously infinite samples represents a prohibitive cost, so from an algorithmic perspective, what is a sufficient number of samples required for minimal error introduced by sampling and still gain insights from the distributions provided by the sampling?

From a computational cost point of view, in abstract this approach has some some apparent advantages. For instance, the calculation $x_A * S_{AB}$ should be considerably cheaper than using W, because S is binary. The biggest cost of the sampling here would be that modern NN hardware technologies, such as GPUs and systolic arrays, are not well-configured for a rapid *in situ* sampling (generating a series of different S matrices for each sample); instead each S would have to be generated from W on at each sampling epoch conventionally, which is a very costly step. However, alternative technologies, such as Sandia's COINFLIPS approach that uses stochastic behavior of magnetic tunnel juncitions, may have the ability to provide this sampling natively.

3 Does sampling a neural network really work

3.1 MNIST network training

Several simple MNIST-trained networks were explored using Keras. The model described here was a simple feedforward network with 784-200-10, with dropout in the hidden layer.

The model was trained for weights to be constrained between 0.0 and 1.0, initialized uniformly between 0.25 and 0.75. The hidden layer had a relu activation, and the output layer was softmax.

The network was trained for 200 epochs, with a batch size of 80, and achieved a validation set accuracy of 0.9625 and a test set accuracy of 0.9535.

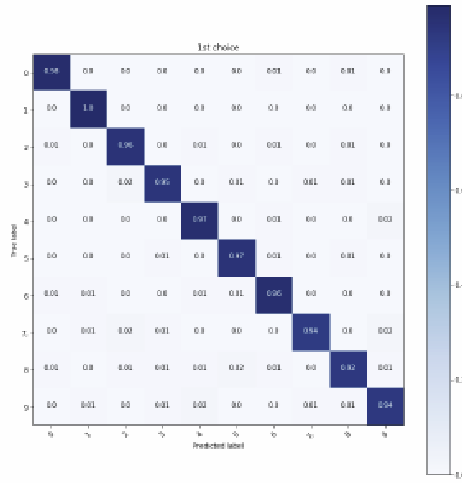


Figure 1: Confusion matrix of deterministic network accuracy

Figure 1 shows the confusion matrix of the originally trained deterministic network.

3.2 Sampling experiment

Sampling was performed by changing the weight matrices W between the first and hidden and hidden and output layers to binary S matrices with each element being 0 or 1 based on a probability draw of $s_{ab} = 1$ with $P = w_{ab}$. There were 1000 separate sampling experiments. The bias weights were unchanged.

Each sample network, now no longer using W but rather the sampled S, had considerably worse performance on MNIST when predicting the test set. Most prediction accuracies were roughly 0.3, with a few as low as 0.2 and a few as high as 0.5. **We are not concerned with the predictive performance of individual samples, however.**

To aggregate the sample performance, we combined the predictions of all of the samples. We consider that each sample of the network basically is a 'vote' of the test data point's appropriate class. In MNIST, this would result in a given data point having noisy votes from each sample. For instance, a

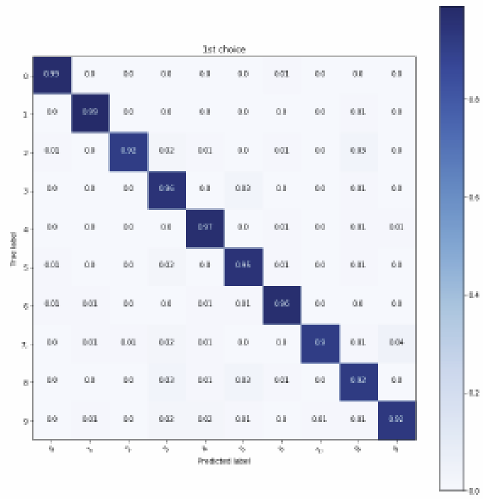


Figure 2: Confusion matrix of first votes

true '4' may get votes for each of the digits (due to vagaries of the synapse sampling), but we would expect most votes to be for 4, while the second most votes would be for perhaps 9, and relatively few votes for something like a 2.

If we consider each of the sampled network get a vote for test data point predictions, we then observe a much stronger performance. As shown below in the confusion matrix of test datapoints first place votes (Figure 2), the vast majority (approaching original training set accuracy) of digits were voted to be appropriately classified.

This is promising, given that the network was not particularly optimized for any form of synaptic sampling (aside from using dropout) and the relatively low number of samples (only 1000).

Sampling here asymptotically appears to approach the test accuracy of the original network (0.9535). After 1000 epochs, the sampling accuracy is 0.9478), and the marginal benefit of additional samples appears to saturate around 300 or 400 samples (Figure 3)

More interesting though is looking at the 2nd place votes (Figure 4). The 2nd place ranking of votes tells us the category, aside from the winner, that the most samples had predicted was the category (note: these 2nd place votes aren't the 2nd choice of a given sample, but rather the 1st choice of

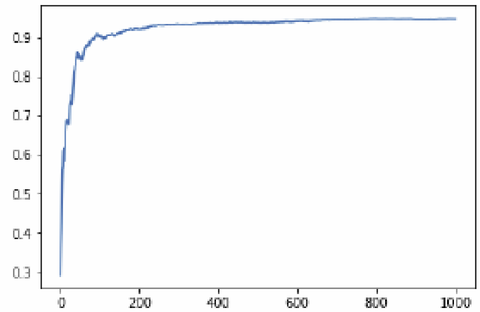


Figure 3: First choice accuracy as samples increase

many of the samples).

This is notable because when we look at the off-diagonal terms, we see how the networks that 'got it wrong' made errors. As would be expected from MNIST intuition, many 4 and 7 data points had a significant number of samples that classified them as a '9'. Similar confusion occurred between 2 and 3. Some of these observations were not noticeable in the first choice errors. For instance, the 2nd choice of many 1s was 7, but there were very few 1s misclassified from the start.

4 The impact of randomness precision on sampling classification

One of the primary goals of COINFLIPS is to control the randomness at the device level, making stochasticity a universal resource. However, while we consider here what we can do a large number of weighted coins, we must also consier that the weighting of that coin likely cannot be arbitrarily set. Rather, the weight will have some precision, and that precision will undoubtedly be lower than the floating point precision typically used in ANN weights.

Here, we repeat the experiments from above, but in a slightly different network, using 400 hidden units (Figure 5) and 1000 sampling epochs. We can see that the 2nd choice sampling plot in Figure 5 is similar to that seen in Figure 4; which is not obvious, since the structure of the incorrect choices that primarily constitute the 2nd choices was not part of the training labels.

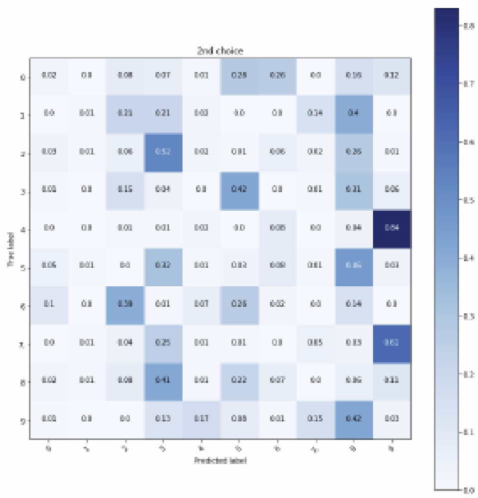


Figure 4: Confusion matrix of second votes

This suggests that the structure of the sampling distributions is inherent derives from the data itself.

Here, we see that with no constraints on sampling precision, the network can get very close (0.9606) to the test accuracy (0.9644).

We then repeated the same sampling process, but instead of comparing the weights to an computer-precision uniform random number, we instead compared to a random number at either 8-bit or 6-bit precision.

Currently, 1000 samples were run for each of these lower-precision cases, and it can be seen that the accuracy possibly does not fully saturate the performance for the 6-bit case (Figure 6). But while lower precision (6-bit) does have a noticeable impact on accuracy, the overall trends of the results are promising, and the 8-bit case does not show any real impairment of precision. In particular, this is showing that considerably lower precision networks (6-bit) still maintain some of their classification ability. While more work is required, the shape of the approximation curve suggests that it is possible a significant part of the remaining error can be mitigated by adding more samples.

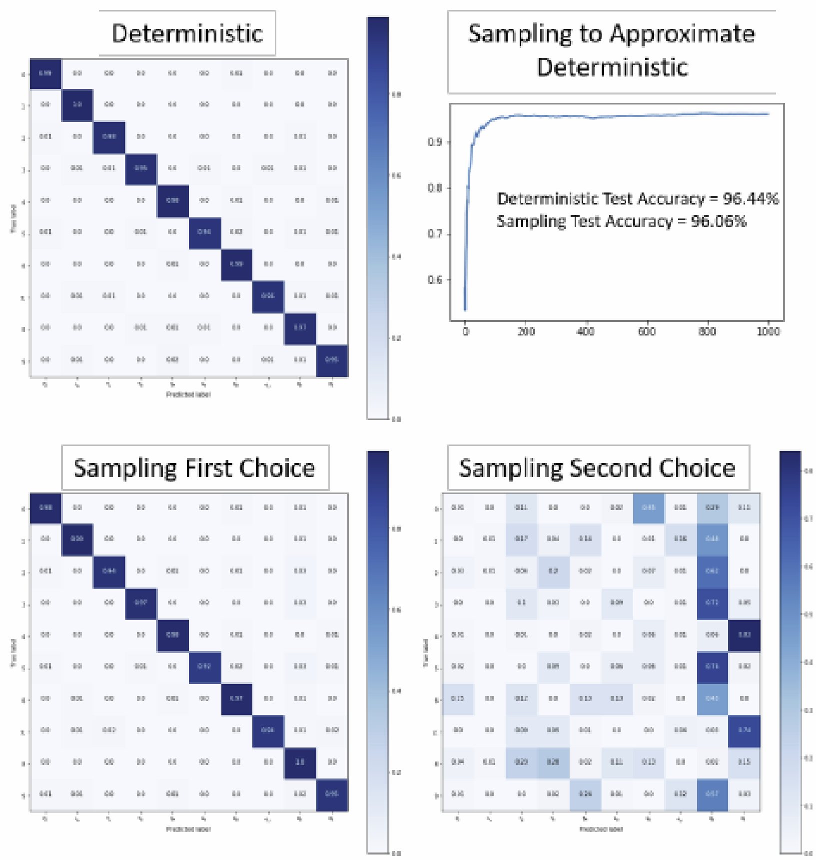


Figure 5: 784-400-10 Sampling Approximation

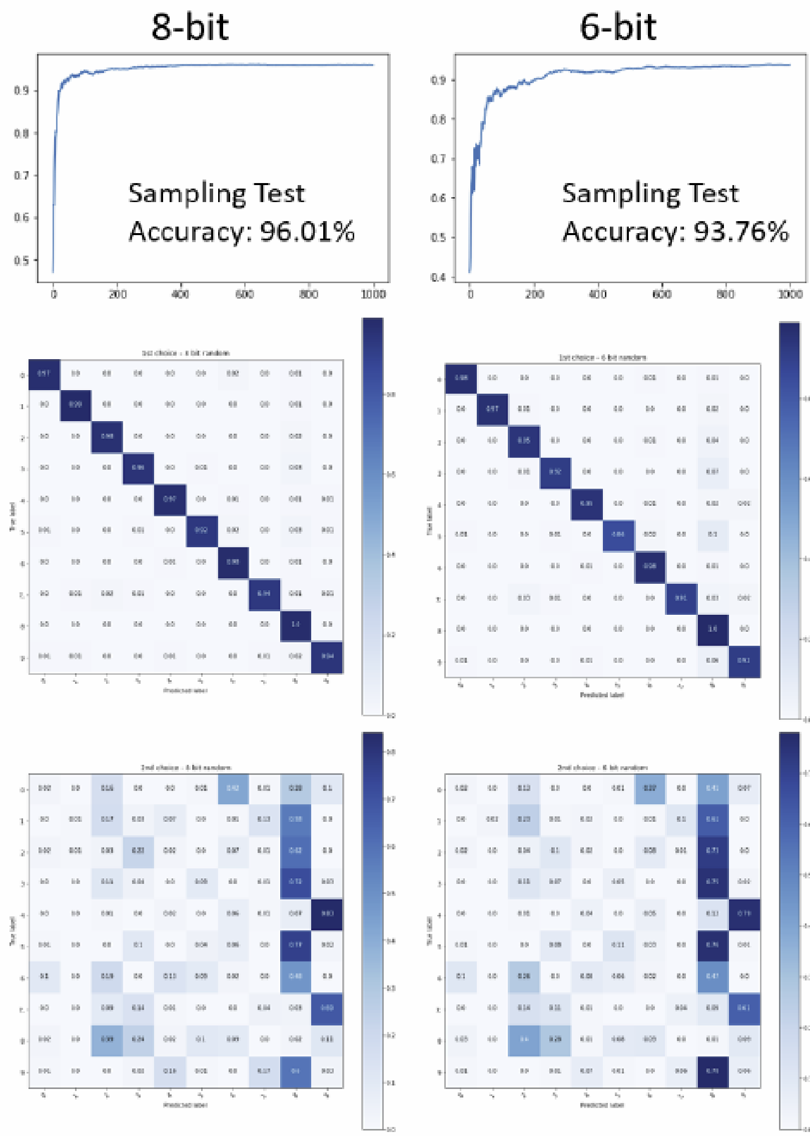


Figure 6: 784-400-10 8-bit and 6-bit Sampling Approximation

Figure 7: Examples of wrong classifications and the sample votes

4.1 Examples of mistakes

We can look at a few specific examples of MNIST examples that are misclassified. As can be seen, there are a number of difficult (for small feed forward networks) examples that are misclassified. As can be seen by these examples, the classification that receives the second most votes is often the correct one (Figure 7).

4.2 Comparable results are observed within Fashion MNIST dataset

A limitation of MNIST as a data set is that the overall performance of even simple networks such as this begins to encounter ceiling effects. The Fashion MNIST data set was developed to replicate the overall task structure of MNIST while making the task itself more difficult.

We repeated the same experiments as above using Fashion MNIST, observing largely comparable results. The deterministic Fashion MNIST network achieved an accuracy of 83.13%, while the sampling of the Fashion MNIST network was able to reach 80.99%. Importantly, the precision preserving effect seen in the MNIST study was seen in Fashion MNIST as well (Figure 8). While there was considerably more uncertainty in this data set, the distribution of 2nd choices was largely preserved across precision levels.

5 Conclusion

This is just a first step, but these results support the idea that sampling can be performed on ANNs, that the approach is not overly prohibitive (can achieve near deterministic accuracy with 1000 samples), and appear to offer

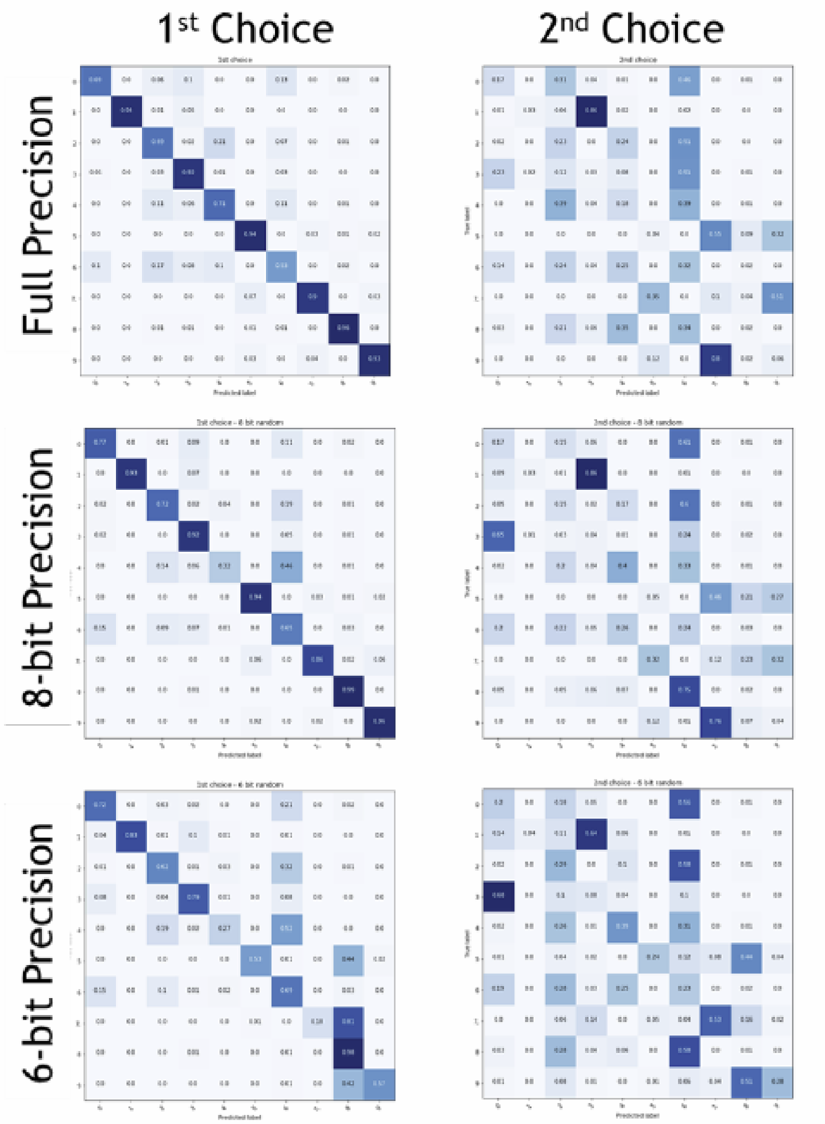


Figure 8: Fashion MNIST networks show similar ability to preserve distributions despite lower precision of random numbers

something beyond a straight deterministic solution in looking a distributions of 'sample votes'.

How this will be extended to convolutional networks is an open question. Specifically, do we sample each kernel once? Or for every use? Other networks will face similar questions.

Aside from the applicability of this approach to more complex networks, an important question is what do these distributions represent? As can be seen from the data here, the similarity between the confusion matrices of 2nd choices in figures 4 and 5 and 6 are indicative that there is some underlying structure that this approach is identifying. Thus, while it is not immediately obvious what a 2nd place tally actually represents (in the approach, a 2nd place tally actually means which digit got the 2nd most 'first place votes' of all the samples), it does suggest that there are certain relationships in the MNIST data that are reliably seen as 2nd choices. This is not a trivial observation - the networks were only trained on the diagonal (the labels represent the 'correct' choices), the off-diagonal terms are all equivalently 'wrong'. That being said, the different networks appear to be 'wrong' in the same ways, which appears to be extracted by the sampling process.

To get to this, it will likely be necessary to intentionally confound the network and present the networks with data that is, in some way, a combination of two different digits. For instance, if we were to blend a 1 and 7, would the sampling process provide a distribution of sampled categorization that is reflective of this pairwise uncertainty? In this case, the ideal answer may actually be half of the samples answering 1 and half answering 7.

Finally, the lower precision examination here provides some encouragement that the approach is amenable for hardware acceleration if ubiquitous random numbers can be generated. The COINFLIPS idea promises ubiquitous random numbers, but almost certainly the precision of these numbers will not be particularly high, especially in systems at large scale. While the 8-bit and 6-bit studies had increased error, both were run with relatively few samples. Further, the distributions of 2nd choices still had the rough structure as the full precision network. This is encouraging and suggests that this approach may be useful in the long-run.