PolygraphMR: Enhancing the Reliability and Dependability of CNNs

Salar Latifi
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
salar@umich.edu

Babak Zamirai
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
zamirai@umich.edu

Scott Mahlke
Computer Science and Engineering
University of Michigan
Ann Arbor, USA
mahlke@umich.edu

***Abstract*—Deep neural networks (DNNs) are now starting to emerge in mission critical applications including autonomous vehicles and precision medicine. An important question is the dependability of DNNs and trustworthiness of their predictions. Considering the irreparable damage that can be caused by mispredictions, assessment of their potential misbehavior is necessary for safe deployment. In this paper, we first show the deficiency of current confidence-based methods as reliability measurement, and assess the effectiveness of traditional architecture reliability methods such as modular redundancy (MR). Then, we propose PolygraphMR and show that the combination of input preprocessing, smarter decision policies, and inclusion of prediction confidences can substantially improve the effectiveness of MR for DNNs. Next, we show how to prohibit explosive growth in the cost of MR by the help of reduced-precision designs and staged activations. Across six benchmarks, PolygraphMR detects an average of 33.5% of the baseline mispredictions with less than $2\times$ overhead.**

***Keywords*-Reliability, Machine vision, Computer performance**

I. Introduction

There is no doubt that Deep Neural Networks (DNN) have revolutionized many domains of applications including computer vision [1], [2], natural language processing [3], speech recognition [4] and handwriting recognition [5]. More and more products and services such as smart speakers, mobile photography, and social networks are integrating these algorithms to facilitate and enhance their usability and functionality. Many different types of DNNs are proposed each targeting different kinds of tasks, for instance, recurrent neural networks (RNNs) are available for tasks like speech recognition, or convolutional neural networks (CNNs) are well-established for image classification problems. Our target in this paper are CNNs for image classification tasks.

CNNs are now moving from non-critical tasks such as gaming and labeling personal photos to mission critical tasks such as pedestrian identification for autonomous vehicles [6], steering commands generation for self-driving cars [7], and patient diagnoses with precision medicine [8], [9]. With mission critical tasks, incorrect answers can be disastrous. While CNN accuracies will continue to rise, robust and reliable CNNs must be realized with imprecise networks. Furthermore, CNNs may never achieve acceptable accuracies for mission critical tasks due to inherent limitations of their mathematical models. Constraints on computation, storage, and energy consumption may also place practical limits on growing CNN sizes, thereby limiting accuracy particularly for energy-constrained environments.

Despite the widespread use of deep learning, there are few metrics to determine the reliability of predictions made by CNNs. When a CNN assigns a label for an input image, there is not any established solution to determine correctness of the prediction and tell apart the correct ones from unreliable wrong answers. This phenomenon leads to an uncertain and unreliable environment when deploying CNN algorithms.

The most apparent solution to this problem is to design networks with higher accuracy. Considering the recent trends in the ImageNet image classification task [10], deeper models with more layers and parameters greatly improve the accuracy of these CNNs [11]–[13]. But, unless we have networks with 100% accuracy, which may not be possible to achieve, this solution is not sufficient by itself. Considering the vital demand for a practical solution, alternative approaches are necessary to assure reliable operation of CNNs.

In prior works [14]–[16], it is argued that the output of the last layer (Softmax) in CNNs can be used as a confidence meter for the network result. The output of the Softmax layer is a vector with size equal to the number of classes in the classification problem, which computes exponentially normalized values of the final fully-connected layer outputs. The final network prediction is the class number with the maximum value in this vector. Therefore, it is possible to interpret the vector values as the probability of assigning the input to the corresponding class, and if the probability value of predicted class is higher, we can make the statement that network is more confident about the generated result. We investigated the use of network confidence for reliability and show that DNNs produce substantial numbers of high-confident wrong answers, thus this metric does not solve the reliability problem by itself (see Section II-B).

The machine learning community proposed network calibration [17] to improve confidence metric credibility. According to network calibration, the reason for high-confidence wrong answers comes from two sources: miscalibration of the CNNs; and, confidence values are not well-

correlated with the actual accuracy of predictions [16]. In other words, if a CNN generates an output which has a confidence of 90%, we cannot make the statement that the probability of answer being correct is also 90%. Network calibration tries to solve this problem by correlating the confidence values with accuracy. Unfortunately, we demonstrate that even with well-calibrated networks, using confidence as a reliability metric still results in considerable mispredictions with high confidence (see Section IV-E).

In this paper, our goal is to develop a practical method to design and realize systems of CNNs that can increase robustness and reliability of classification results with imprecise and currently available CNNs as the building blocks. The goal is not to increase accuracy but rather focus on the dependability of results. PolygraphMR uses input preprocessing techniques to develop different variations of a CNN and combines them as a modular redundant (MR) system of heterogeneous CNNs. It also employs a tunable decision policy engine that uses outputs of the networks and user's reliability demands to make decisions on the trustworthiness of each prediction. However, MR systems are notoriously expensive in terms of area and energy consumption. Thus, the footprint of each CNN variant in the MR system is reduced by scaling down the data precision and intelligently staging activations of individual CNNs so that most of the time only a subset of the MR system is active.

PolygraphMR enhances reliability by leveraging variations in behavior of each CNN that is trained in different circumstances and environments. It takes advantage of behavior diversity to detect symptoms of unreliability from the predictions of different CNN variants. A systematic approach is used to develop an efficient PolygraphMR system for any benchmark using the initial CNN as the basic building block. The functionality of PolygraphMR is orthogonal to the accuracy and topology of baseline CNN and can be applied to the future networks with higher accuracy levels for further reliability enhancement.

This paper makes the following contributions:

- We demonstrate that high confidence, but wrong answers are a problem for most CNNs. We also show that current solutions such as using a confidence threshold or calibrating the network confidence do not satisfactorily solve the reliability problem.
- We introduce PolygraphMR, a heterogeneous MR system of CNNs that detects unreliable predictions by recognizing the symptoms of unreliability in the behavior variation among the constituent CNNs. Behavior diversity is synthesized by using a set of simple image preprocessing techniques for training/inference.
- We eliminate a large fraction of traditional MR overheads by scaling down the data precision of individual CNNs and deploying a resource-aware decision engine to activate only a subset of CNNs for each inference. We show that more aggressive data precision scaling without sacrificing prediction accuracy is possible with a PolygraphMR system than for a standalone CNN.
- We evaluate PolygraphMR system across three well-known image classification datasets and six CNNs and show that it is capable of detecting an average of 40.8% of mispredictions in a non-constrained resource environment, or 33.5% of mispredictions with less than 86.5% overhead on energy and latency.

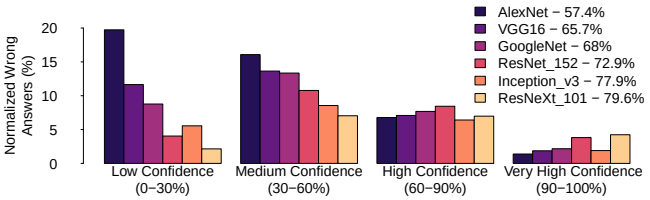


Figure 1: Histogram of normalized wrong answers generated by AlexNet [11], VGG16 [19], GoogleNet [20], ResNet_152 [12], Inception_V3 [21], and ResNeXt_101 [22].

II. Motivation

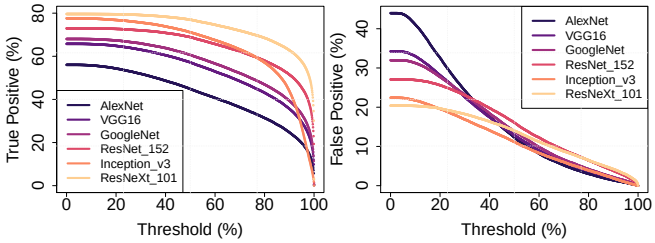
A. High-Confidence Wrong Answers

In order to better understand the reliability problem of current CNNs, we analyze errors of the six well-known networks for ImageNet dataset [18]. Benchmarks and their top-1 accuracies for the validation set are presented in Figure 1. The output of Softmax layer is used as the probability/confidence values of labels as suggested by previous works [14], [16]. Figure 1 presents the distribution of wrong predictions made by CNNs across the entire validation set. To ease the analysis, wrong answers are grouped into four categories based on their prediction probabilities: low $(0-30\%)$, medium $(30-60\%)$, high $(60-90\%)$ and very high $(90-100\%)$ confidence. All bars are normalized by the total number of samples in the validation set, so the distributions can be compared to each other.

It is clear that the low and medium confidence wrong answers are the largest for most CNNs, which is intuitive. But, nearly 10% of the answers are wrong with high or very high confidence for each CNN. From a reliability perspective, 10% high confidence wrong answers is quite large. Figure 1 also exposes a more subtle, but concerning trend. As the CNNs become more accurate, severity of problem increases and there are a higher fraction of high confidence mispredictions. This shows that the higher accuracy is mainly coming from correctly predicting low confidence wrong answers of less accurate CNNs, and overall, confidence values for majority of predictions, whether correct or wrong, are shifted higher which makes them less reliable.

B. Limitations of the Confidence Metric

To explore the use of confidence as a reliability metric more deeply, one approach is to choose a minimum threshold for the prediction probability in order to consider it reliable.



(a) True Positives (b) False Positives

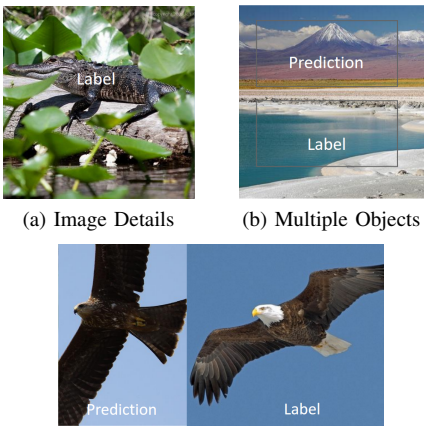
Figure 2: Effect of the probability threshold on the network predictions. (a) Distribution of true positives over threshold value. (b) Distribution of false positives over threshold value.

When the prediction probability falls below the threshold, the CNN output is unreliable. Figure 2 demonstrates the rate of undetected mispredictions, or False positives (FP), and correct predictions, or True positives (TP), as a function of the confidence threshold. At a threshold of 0, none of the predictions are affected and the rate of FP and TP matches the original accuracy. As the threshold increases, the FP rate is decreased but at the same time, TP rate goes down as a portion of correct predictions are lost due to their low confidence. As depicted in Figure 2a, the change in TP rates for different CNNs tends to be similar, thereby maintaining a relatively constant difference in TP values regardless of threshold. Conversely, Figure 2b demonstrates the higher vulnerability of more accurate CNNs to the high confident wrong answers. Although the FP rate is initially lower in more accurate CNNs, the curves cross and the less accurate CNNs get lower FP rates at higher thresholds. This result again reinforces a somewhat counter-intuitive result that as accuracies go up, it is more difficult to eliminate the FPs and overall we have more high-confident wrong answers.

C. Misclassification Analysis

To get a better understanding of CNN behavior, we analyzed the wrong predictions of AlexNet over the validation set of ImageNet. The highest confidence wrong answers, i.e, mispredictions with confidence of 90% or more, which corresponds to about the top 5% of wrong answers, were manually examined to determine if any trends were apparent. Figure 3 summarizes the top 3 characteristics. The first characteristic is having poor image detail which includes obstruction, obfuscation, blur, etc. Figure 3a presents an example where the crocodile is obstructed by leaves in the foreground. The second characteristic is to have multiple objects in the image as shown by Figure 3b. This picture contains two objects, a seashore in the front and a mountain in the back. In this example, network incorrectly predicted the mountain whereas seashore was the correct label. Finally, the third characteristic is the similarity between classes as shown in Figure 3c. The right image is labeled as bald eagle in the dataset, while it is mispredicted as a kite, and the left image is a sample of kite class, which shows its similarity.

The critical problem is not that these images are mis-



(a) Image Details (b) Multiple Objects

(c) Class Similarity

Figure 3: Misclassification analysis on ImageNet dataset.

predicted, but rather the wrong prediction is made with very high confidence (e.g., over confident predictions). Humans may also classify these images incorrectly, but in contrast would likely have lower confidence due to the image characteristics. This points out one of the limitations of machine learning that the underlying mathematical models do not differentiate hard versus easy to classify images nor utilize confidence as an input to the training process. Rather, training designates a ground truth class for each sample. During training, weights of the network are continuously updated until the probability outputs are converged toward the ground truths. So, the network is forced to get the probability of the output corresponding to the label class number to 100%, making it more sensitive and reducing the generality of the trained model [23]. And as a result, the network ends up making over confident predictions.

We hypothesize that this limitation can be alleviated by creating a system of networks that provides both multiplicity and diversity. Multiplicity enables multiple independent predictions on the same input and can be used to adjust confidence based on agreement of answers. And diversity further enhances multiplicity by differentiating individual learners, thus increasing overall comprehensiveness of prediction space. We believe combination of these two characteristics can help our system to perform better at approaching and resolving more demanding inputs. The challenges are then to systematically create heterogeneous systems of multiple networks so the user is not burdened with this task and mitigate the multiplicative factors of energy/cost that deploying multiple networks will seemingly require. Evaluating the merits of this hypothesis while overcoming these challenges is the focus of PolygraphMR and the rest of this paper.

III. PolygraphMR

A. Overall Design

PolygraphMR (**PGMR**) uses available imprecise CNNs as building blocks and leverages behavior diversity between them as symptoms of unreliability and likelihood of unreliable predictions. Figure 4 shows an overview of the system

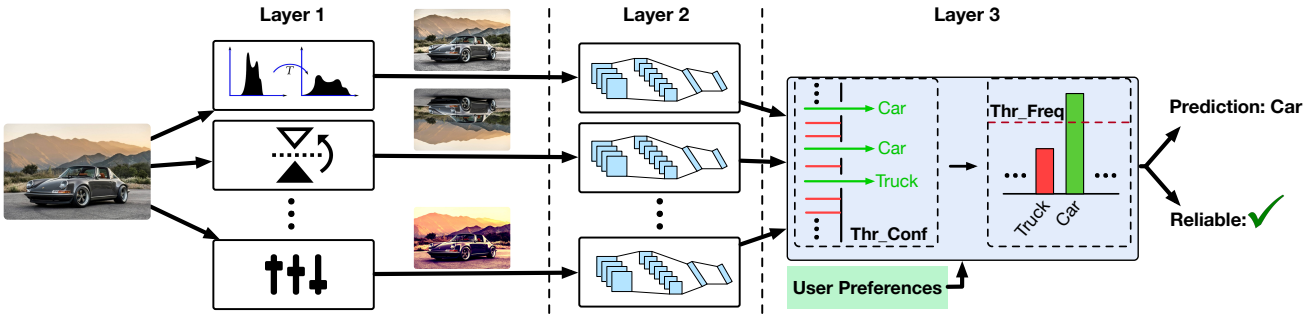


Figure 4: General design of a PolygraphMR system: Layer 1 is responsible to preprocess input image and inject diversity to system, layer 2 provides redundancy by making prediction on individual inputs, and layer 3 generates final prediction and decide on output reliability.

for single discrete inputs, still images in the illustration. The design consists of three layers: preprocessing units (Section III-B), modular networks (Section III-C), and decision making engine (Section III-E).

For inference, the system operates as a group of heterogeneous DNNs (Layer 2) that are driven by a diverse set of real and synthetic inputs crafted from the original input by the available preprocessors (Layer 1). The goal of creating such a wide input diversity is to provide more information in order to render more confident predictions. Outputs of the heterogeneous group are analyzed to determine the final prediction and also whether or not the answer is reliable (Layer 3). The decision making portion is configurable and users can specify the target reliability for the system. The final output of this system could land in one of the following three domains: True Positives (TP) which are correct and reliable answers, False Positives (FP) which are undetected mispredictions, and Unreliable answers which are composed of detected wrong answers (false negatives) and correct answers that are undesirably marked as unreliable (true negatives). Our reliability goal in this paper is to reduce FPs as much as possible by marking them as unreliable predictions, while keeping the desired TPs unchanged.

B. Layer 1: Pool of Preprocessors

Traditionally, diversity in CNNs is created by random initialization of weights in the training phase. However, as expected, the amount of diversity is very limited as will be shown in Section III-C. Instead, we turn to prior work that has studied image preprocessors and shown they are helpful in improving the overall accuracy of CNNs [24]–[26]. Our goal of preprocessing is instead to create a group of CNNs with diversity in behavior. We hypothesize that diversity will help us identify inputs where the prediction should be treated as unreliable due to behavior variation across the CNNs.

Each CNN in layer 2 will be fed by transformed images generated by one of the preprocessors. We can use simple linear transformations like flipping, or more complex nonlinear functions like histogram equalization or contrast normalization as preprocessors. Each of these preprocessors introduces a different level of diversity to the system depending on the dataset or functionality of the preprocessor itself. We examine the effectiveness of each preprocessor and compare them together in Section III-G. Table I presents the preprocessors that are used across our benchmarks in Section 9. Among these preprocessors, FlipX and FlipY are used more frequently. On the other hand, ImAdj is used only by one of the benchmarks. Overall, we observed that the preprocessors which preserve the vital features of the inputs while providing sufficient diversity to the system, are more frequently used across different datasets. Whereas a preprocessor like ImAdj, which heavily modifies the input features, like pixel colors, is less useful.

Table I: Image preprocessors and their functionality.

Preprocessor	Functionality
AdHist	Locally adjusts image intensities to enhance contrast
ConNorm	Locally normalizes image contrast
FlipX	Flips image in the horizontal axis
FlipY	Flips image in the vertical axis
Gamma	Gamma correction, controls the overall brightness
Hist	Adjusts image intensities to enhance contrast
ImAdj	Maps image intensity values to a new range

C. Layer 2: Heterogeneous Modular Redundancy

The base of layer 2 is Modular Redundancy (MR), which is well recognized as a standard solution for building mission critical computer systems [27]. With this approach, multiple copies of the unreliable module are instantiated and activated with a majority vote taken to decide the final answer. If the modules operate correctly most of the time, then the majority are likely correct for any single input. For probabilistic models including CNNs, the goal of using MR is separating reliable and unreliable answers rather than increasing the accuracy. Therefore, when the CNNs agree, the output is labeled as reliable and when they disagree then unreliable. To assess the success of MR, FP rates are measured.

We extend the traditional MR design with two modifications to account for the probabilistic behavior of CNNs:

1) We change the decision policy from majority voting to require a specific number of networks to agree on the final prediction. We call this number the *frequency threshold* (Thr_Freq).

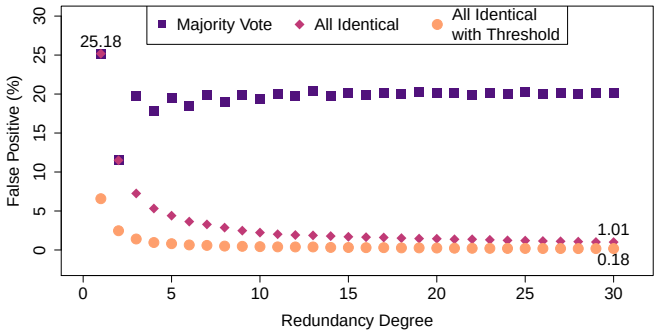


Figure 5: Different modular redundancy methods applied to ConvNet on CIFAR10.

2) We include a confidence threshold for accepting the prediction result from each network. If the confidence of a specific prediction is below the threshold, it will be neglected. For the rest of the paper, we call this value the *confidence threshold* (Thr_Conf).

To evaluate traditional MR, we run an experiment using ConvNet on the CIFAR-10 dataset [28]. This dataset contains 10k images which are classified into ten categories (1k images per category) with a baseline accuracy of 74.7%. The degree of MR varies from 2 to 30. MR networks are created by instantiating n copies of the baseline CNN, randomizing the starting weights, and training each on the original dataset. Each CNN ends with different weights and thus behaves differently. Figure 5 shows the impact of redundancy degree on the number of wrong answers predicted by the MR system with different decision policies: 1) Traditional MR with majority voting (Majority Vote), 2) MR with a Thr_Freq equal to the number of CNNs, which will require all networks to predict the same label and is the most restrictive threshold (All identical), 3) the previous MR design plus a Thr_Conf of 75% (All identical with Threshold). The Thr_Conf is chosen somewhat arbitrarily, but which corresponds to a relatively high confidence threshold. Note that for the design with majority voting if two classes share the same highest frequency, the result is considered unreliable.

From Figure 5, majority voting does not provide a significant decrease in FPs regardless of the redundancy degree. The FP rate flattens at about 20%, after starting at 25.2% with a single CNN. MR with the Thr_Freq is much more successful, reducing the FP rate down to 1%. MR with both Thr_Freq and Thr_Conf decreases the FP rate even further to 0.18%. However, the latter two solutions have an undesirable side effect of substantially decreasing the number of TPs. For example with the All Identical method, to achieve a 1% FP rate, TPs reduce from 74.7% with a single network to 40.4% because a large number of correct predictions are considered unreliable.

We make three conclusions based on these results:

First, traditional MR is incapable of effectively reducing the FP rate regardless of the redundancy degree, whereas they are shown to be practical while applied to other computer reliability problems such as transient faults [29], [30]. The reason is that with traditional computer systems, execution of applications is flawless in a fault-free setting and faults are relatively rare. However, CNNs are inherently faulty regardless of the hardware, and errors are much more common due to the inherent inaccuracy of the mathematical models. This results in a significant disagreement between individual CNNs and poor results.

Second, from the majority voting results, the effect of behavior diversity reduces the FP rate by 5% without any loss of TPs. Conversely, the all identical voting has much larger drops in FPs, but loses too many TPs. Thus, it is important to have less restrictive voting mechanisms like majority while at the same time injecting larger amounts of diversity than simply random initial weights, which led to our decision to use input preprocessing.

And finally, single CNNs are expensive in terms of computation, storage, and energy consumption. Thus, the multiplicative cost increase of CNNs in an MR system could easily become infeasible. Thus, we need to deploy an resource-aware implementation of our MR system.

D. Resource-aware MR (RAMR)

Due to inherent redundancy in Layer 2, computation of PolygraphMR introduces new energy and latency overheads per inference. To mitigate a portion of these overheads, we explore narrow-precision floating-point representation for each CNN in the system.

We follow a similar implementation introduced in [31], [32] and reduce the overhead of data transfer by precision reduction. We assume all weights and intermediate values are using a lower precision. Hence, it is possible to pack them together during both on-chip and off-chip data transfer. As a result, the reduced traffic on memory hierarchy leads to higher utilization of compute units and higher performance.

Although there is an opportunity to reduce the cost of PolygraphMR by using narrow-precision computation, we should acknowledge that this solution is not unique to our system and can be applied to any CNN for energy saving purposes. To analyze the effect of lower precision, we run an experiment using AlexNet [28] on the ImageNet [18] dataset. The goal is to compare the degree of precision reduction on a PolygraphMR system with a baseline of an individual AlexNet. We hope that the combination of diverse CNNs in our system would be more resilient against the negative effects of lower precision on accuracy. Therefore, we would be able to further reduce the precision of each individual CNN in comparison to the baseline.

Figure 6 presents the accuracy of both PolygraphMR and the baseline CNN with respect to the precision they are using. We focus on maintaining the accuracy level of the baseline CNN, which is 57.4% for AlexNet, and investigate the possibility of achieving this accuracy at each precision

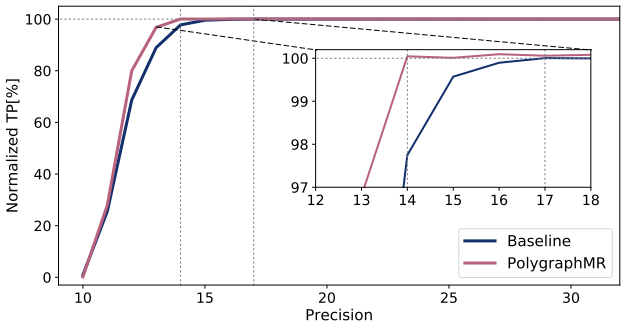


Figure 6: Effect of precision reduction on original and PolygraphMR system using AlexNet.

level. At the first glance, both designs seem to be responding equally to the reduction of precision. But in a closer look, we can see that the baseline AlexNet starts to lose its accuracy passing 17 bits precision level. On the other hand, PolygraphMR can still tolerate lower precision levels of up to 14 bits. In fact, the accuracy of each individual CNN in PolygraphMR also follows a similar trend to the baseline AlexNet, but combining their predictions and then making decision performs similar to ensembles and compensates for the individual accuracy drop. As a result, we can further reduce the precision on PolygraphMR system and mitigate a portion of the multiplicative energy and latency overhead.

E. Layer 3: Decision Engine

The last layer of PolygraphMR is responsible for collecting the outputs from layer 2, generating the final prediction of the system, and determining whether or not the prediction is reliable. This happens in two steps during the system inference phase. First, the decision engine compares the probability vectors generated by the softmax layer of each network, with the designated Thr_Conf value and forms a histogram of acceptable votes for each class. As a result, all labels in individual output vectors, which meet threshold restrictions, are recorded in the histogram. Next, the decision engine reports the class label with the highest frequency as the final prediction of the system and reports the reliability of the label by comparing the respective frequency with the preselected Thr_Freq.

The appropriate values for Thr_Freq and Thr_Conf are determined after training the MR networks and during an offline profiling stage. First, the value space for the set of thresholds is swept, and the respective TP and FP rates of the design points over the validation dataset is recorded. This process is not time consuming and has a negligible overhead compared to the actual training of the MR networks. Next, a Pareto frontier for the threshold values, which maximizes the TPs and minimizes the respective FPs, is formed. And finally, a set of Thr_Conf and Thr_Freq values is selected from the Pareto frontier based on the user demands, which might be a specific TP or FP limit.

The threshold values selected in the profiling stage, will

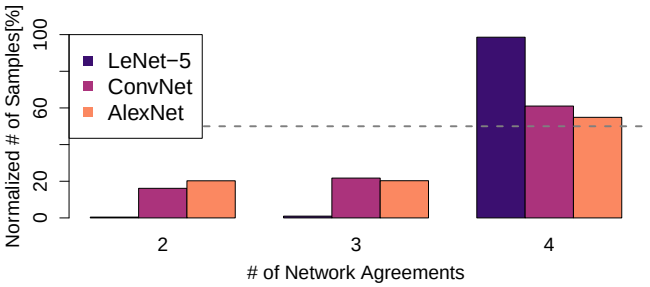


Figure 7: Histogram of prediction agreements in a system with 4 CNNs, profiled on LeNet-5, ConvNet, and AlexNet.

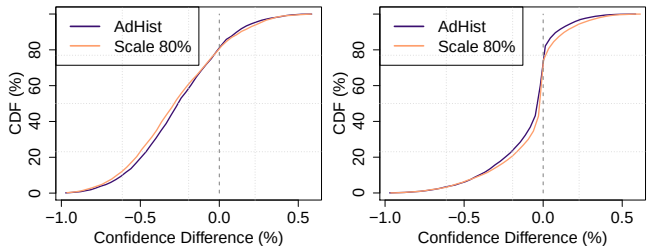
be fixed during the inference phase of the system. But, if the user demands are updated at any point, a new set of threshold values can be selected from the Pareto frontier to meet the new requirements.

F. Resource-aware Decision Engine (RADE)

To further reduce the performance overhead of Layer 2, we deploy a resource-aware decision policy. Unlike the traditional MR with majority voting which requires all prediction outputs from every network to be present in order to make the final decision, PolygraphMR can decide on the reliability of the prediction if a subset of the networks provide the same label with a minimal confidence.

To analyze the number of networks that need to be activated, we run an experiment using a PolygraphMR with four networks on three benchmarks: LeNet5 [33] on MNIST [34], ConvNet [35] on CIFAR-10 [28], and AlexNet [28] on ImageNet [18] dataset. Our goal from this experiment is to gain a general idea about how often we need to activate all networks to get a reliable prediction. We collect the prediction results of each network to see how often they are in agreement with each other. To simplify the experiment, we do not use any Thr_Conf and just gather the top prediction from each network regardless of its confidence. Figure 7 presents histogram of network agreements for our benchmark. The x-axis shows the number of agreements among the four CNNs, and the y-axis represents the corresponding normalized frequency over test samples. We can see that in more than 50% of times, we don't need to activate all the CNNs since their prediction results are in harmony with each other. This gives us the opportunity to activate just a portion of CNNs and reduce the performance overhead. But, the key point is to decide on which network(s) to activate, since there is a possibility that we would face an input where predictions from the selected networks might conflict with each other. An oracle decision engine would be the one which activates the single reliable CNN per input sample. But even if it is possible to design such an engine, it is likely complex and would consume considerable energy.

In order to design a resource efficient decision engine, we use a priority scheme on CNN activations. In this case, we can stage activation by activating a batch of high priority



(a) Wrong predictions. (b) Correct predictions.

Figure 8: Comparing effects of AdHist and Scale 80% on confidence changes with respect to original CNN.

CNNs first to check their predictions, and only continue to execute other CNNs if we are not able to determine the final answer after the first round of invocations. This approach can give us the opportunity to have early detection of TP or unreliable answers, and result in lower energy consumption.

To come up with a priority scheme, we statistically analyze the contribution of each CNN in the system. In other words, we record the frequency of instances that each network provides a correct label to the decision engine over a specific number of test cases during training. Next, we use the measured frequency numbers to give priority to each network. In the inference phase, we first execute the top Thr_Freq networks to see if we can determine the final answer. Next, we move to other networks based on their contribution until we're ready to generate the output. Energy saving results and latency improvement of this decision engine are discussed in Section IV-C.

G. PolygraphMR System Design

We use a two-step procedure to select the best preprocessors and form a PolygraphMR system for each individual application and dataset. First, we compare the relative performance of preprocessors regarding the potential behavior diversity each can introduce. Next, a set of candidate preprocessors is used in a greedy approach to select the final preprocessed networks. As discussed in Section III-B, a wide range of linear and non-linear preprocessors are available to use. But, not all of them provide sufficient behavior diversity to justify their energy overhead. Hence, the first step is to compare preprocessors and select the best ones.

For each input instance, we profile the difference between prediction confidence of the baseline CNN and each preprocessed CNN, called $delta$. The delta values are then used to compare pairs of preprocessors. Figure 8 presents a comparison between AdHist (in which image intensities are locally adjusted to enhance contrast) and Scale 80% (where input image is scaled down and up by 20% to soften noise levels) on ConvNet. The x-axis presents $delta$ values, and y-axis projects the corresponding cumulative distribution. Figure 8a displays the distribution of $delta$ values for inputs that are initially mispredicted by the baseline CNN. As shown, AdHist has a higher probability in negative $deltas$. Even though the difference is relatively small, the likelihood of having lower confidence for mispredicted results with the AdHist is higher. Thus, the probability of getting the same misprediction with AdHist is lower than Scale 80%, and AdHist would be a better preprocessor to introduce behavior diversity for this network. In addition to Figure 8b, Scale 80% has higher probability in negative $deltas$, which means there is a higher chance of getting a lower confidence for samples that are correctly predicted by the baseline. Hence, the probability of predicting the same correct answer is lower compared to AdHist. This comparison shows that the AdHist has a higher potential for introducing behavior diversity to our system. In our experiments, preprocessors listed in Table I are the most frequently selected ones.

The second step is to run an iterative greedy approach on candidate preprocessors to select the final networks for PolygraphMR. First, it starts by selecting a baseline CNN as the first network in layer 2. It also gathers the respective TP and FP rates to be used as baseline. Next, each of the preprocessed CNNs is separately selected and added to the current configuration and reduction in FP rate is recorded. Then, the best preprocessor for the current iteration is added to the final design. We keep iterating through this algorithm until a predefined maximum number of CNNs is reached.

Dataset	CNN	Accuracy	# of Layers	# of Classes
MNIST	LeNet-5 [33]	99.01%	5	10
CIFAR10	ConvNet [35]	74.70%	4	10
	ResNet20 [12]	91.50%	20	10
	DenseNet40 [36]	93.07%	40	10
ImageNet	AlexNet [28]	57.40%	8	1000
	ResNet34 [12]	71.46%	34	1000

Table II: Benchmark set used to evaluate PolygraphMR.

IV. Evaluation

We evaluate PolygraphMR in two stages. First, we focus solely on reliability improvements without considering any performance optimizations in Section IV-B. Next, we apply RAMR and RADE to reduce the overheads in Section IV-C.

A. Methodology

Benchmarks: Three datasets are selected for evaluation: MNIST [34], CIFAR-10 [28], and ImageNet [18] dataset. As for the CNNs, we choose six networks with different ranges of accuracies and topologies to show that our solution is independent from the original network correctness level and architecture. Table II presents the accuracies and specifications of each benchmark.

Comparisons: To evaluate reliability of optimal configurations proposed for each benchmark, two comparisons are made. We compare PolygraphMR with N networks (N_PGMR) with the original baseline network (ORG), and with an MR system composed of N networks (N_MR).

Evaluation Metrics: For the reliability comparison metric, we use FP rate of design points where no desirable correct predictions are lost. Therefore, the FP rates included in the rest of the results section correspond to design points

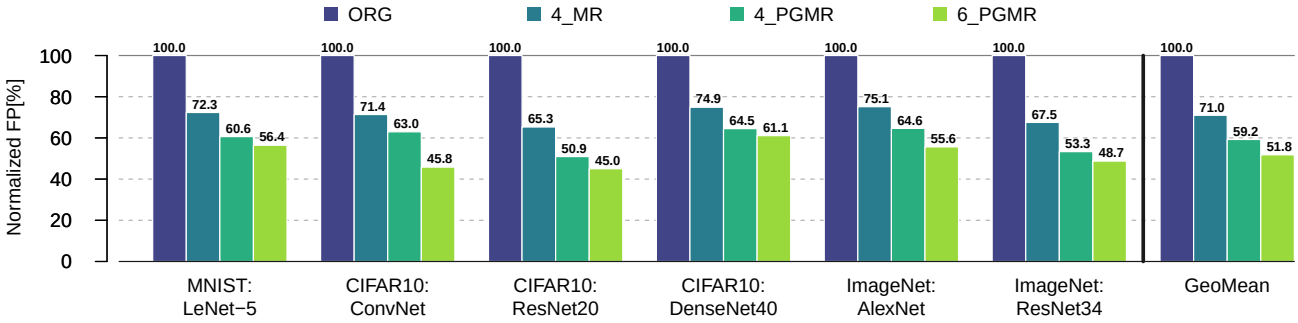


Figure 9: Comparison of normalized FP rate for each design on different benchmarks (TP rate of all design points are matching baseline accuracy level).

Dataset	CNN	Configuration
MNIST	LeNet-5	ORG, ConNorm, FlipX, Gamma ($\gamma = 2$)
CIFAR10	ConvNet	ORG, AdHist, FlipX, FlipY ($\gamma = 2$)
	ResNet20	ORG, FlipX, FlipY, Gamma ($\gamma = 1.5$)
	DenseNet40	ORG, ImAdj, Gamma ($\gamma = 1.5$), Gamma ($\gamma = 2$)
ImageNet	AlexNet	ORG, FlipX, FlipY, Gamma ($\gamma = 2$)
	ResNet34	ORG, FlipX, FlipY, Gamma ($\gamma = 2$)

Table III: 4_PGMR configuration selected for each benchmark (ORG stands for original baseline network).

with normalized TP of 100% of the baseline network. FP rates are also normalized with respect to the ORG FP rate.

We also use latency and energy of original CNNs for each benchmark as our performance measurement baseline.

Preprosseing: We use OpenCV library and MATLAB for preprocessing of the datasets during training and testing.

Reliability Modeling: We use Caffe [37] framework to implement, train and test our CNNs. To evaluate the accuracy of low-precision networks used in RAMR, we modify the Caffe library by replacing default cuDNN framework with our custom CUDA kernels that support variable precision. This enables changing inference precision by truncating values of load and store instructions to the desired settings. We use a unified precision throughout the network and for all layers.

Performance Modeling: Energy and latency of PolygraphMR are measured on a machine with Intel Core i7-5930K CPU and an NVIDIA TITAN X (Pascal) GPU. Inference of each CNN in PolygraphMR is done sequentially, and at the end, decision policy is deployed to get the final result. In this pipeline, preprocessing and CNN inference are executed on GPU, whereas decision policy is based on CPU.

To measure the performance of low-precision CNNs, we model data packing and unpacking in software which is then integrated in our kernels. Next, GPGPUsim v4.0 [38] and GPUWattch v1.0 [39] are used with TITAN X configuration [40] to run the simulation on individual benchmarks.

B. Reliability Results

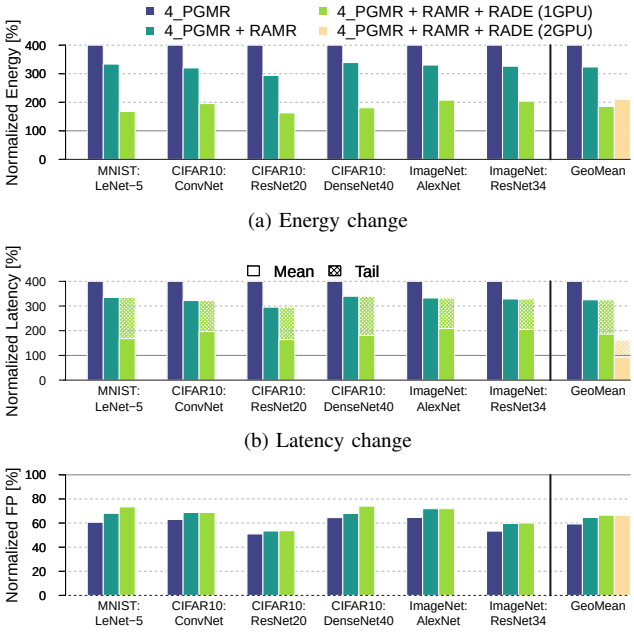
In this section, we assess the reliability results of PolygraphMR without including any side effects of performance optimization. We evaluate two configurations of PolygraphMR, one with four networks (4_PGMR) and another with six networks(6_PGMR) to show the scalability.

Figure 9 shows the evaluation results for all six benchmarks. We compare the normalized FP rate of the PolygraphMR system with other designs. The y-axis displays FP rate which is normalized to the FP rate of the corresponding baseline CNN. All design points also have a normalized TP of 100%. One can see that on average, 4_PGMR can reduce the FP rate of the baseline CNN by 40.8%. This value is also 16.6% lower compared to MR configuration with the same number of CNNs. Table III summarizes the preprocessors selected for each benchmark in 4_PGMR configuration.

Figure 9 also shows that PolygraphMR designs are orthogonal from the baseline accuracy. For example in CIFAR10, three benchmarks with different accuracy and complexity levels are evaluated. We can see that 4_PGMR is successfully reducing FPs in all three benchmarks. The same observation can be made on the ImageNet based benchmarks. In conclusion, PolygraphMR can work in harmony with future more accurate networks to enhance their reliability.

Another interesting point is the selection of FlipX preprocessor in the benchmarks that are evaluated on ImageNet. During the training of AlexNet and ResNet34, samples are randomly flipped and fed to the network to introduce robustness towards the rotation of objects in the image. Hence, there was not any gain expectation by addition of this preprocessor, since we believed that the intended diversity was already introduced to the system during the training. But as the result of final configuration depicts, we still get benefits by deploying FlipX. This shows the sensitivity of the network to minor changes in the input and the explanation for this phenomenon is discussed in Section 3.

Figure 9 also presents FP rates for 6_PGMR configurations. On average, 6_PGMR designs can detect 48.2% of baseline FPs, which is 12.5% improvement over 4_PGMR. Among all benchmarks, ConvNet on CIFAR10 and AlexNet on ImageNet benefit the most from increased diversity by, respectively, detecting 27.3% and 14% more FPs over their 4_PGMR counterparts. But, we observe that for the majority of benchmarks, 4_PGMR provides the sweet



(a) Energy change

(b) Latency change

(c) FP change

Figure 10: Energy, latency, and FP rate trend during cost-oriented optimization.

spot in reliability and cost trade-off. For the rest of paper, 4_PGMR designs are used for further analysis.

C. Energy/Latency Optimizations

To alleviate performance overhead of PolygraphMR, we deploy a two-step optimization procedure. First, we reduce the precision of each individual CNN in the 4_PGMR as discussed in Section III-D. Next, we replace the decision engine of the new system with a resource-aware version.

In our evaluation, we assume that the proposed design is executed on an average hardware setup including only one GPU. This is the worst case scenario which requires PolygraphMR to execute individual networks sequentially. Therefore, the latency and energy overhead will grow linearly with the number of networks. If a more advanced hardware setup with multiple GPUs is available, such as the NVIDIA DRIVE AGX self-driving compute platform equipped with two TensorCore GPUs [41], latency overhead can be scaled correspondingly down. The latencies of the preprocessing and decision engine are also measured, but their overhead is negligible compared to CNN computation, e.g., 2.5% for AlexNet and 0.6% for ResNet34.

4_PGMR + RAMR: For each benchmark, we reduce the precision of both the baseline CNN as well as all CNNs in the 4_PGMR. As shown in Section III-D, the PolygraphMR system is more resilient to precision reduction than a single CNN. Therefore, we can reduce the precision of individual CNNs in the 4_PGMR more aggressively. Figure 11 presents the precision reduction results on AlexNet benchmark. We compare the Pareto frontier of baseline and

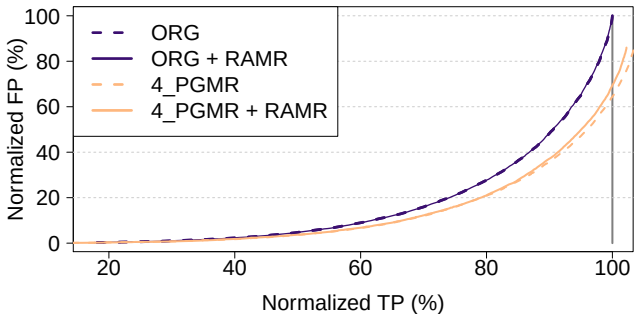


Figure 11: Pareto frontier comparison of precision reduced AlexNet on ImageNet dataset.

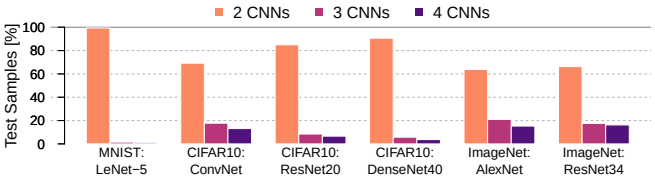


Figure 12: Distribution of number of networks activated in 4_PGMR system over test set of individual benchmarks.

4_PGMR in full precision and reduced precision settings. To get the Pareto frontier of ORG, it is coupled with a confidence threshold. In this figure, the x-axis represents the TP rate and y-axis stands for FP rate, both normalized to corresponding baseline rates. In this experiment, the precision of ORG is reduced to 17bits without any accuracy loss. As for 4_PGMR, precision is further decreased to 14bits again with no accuracy loss. But, as shown in Figure 11, the FP rate of 4_PGMR system is little changed with RAMR, and still offers 28.1% FP detection rate. We observe a similar behavior for other benchmarks and precision of each CNN is further-decreased by two to four bits.

The second bars in Figure 10 summarize the effect of RAMR on energy, latency, and normalized FP rate. On average, we can reduce the energy consumption and latency overhead by 76.5% and 75.0%, respectively. Whereas, FP rate is modestly increased by 5.4%.

4_PGMR + RAMR + RADE: Next, we deploy the resource-aware decision engine described in Section III-F. Figure 12 presents distribution of the number of networks activated by RADE for individual benchmarks over test set. We observe that the majority of samples only require two CNNs to get the prediction. We only need to activate more networks for more demanding and complicated inputs. Figure 12 also shows that benchmarks with a higher accuracy baseline, less frequently require the activation of extra networks.

The third bars in Figure 10 present the energy and latency reductions as well as FP rate changes. By applying both performance optimization, we reduce average energy overhead to 185.5% and normalized average latency to 186.3%. On the other hand, normalized FP rate is increased by 7.2%.

Although RADE is effective on reducing the average latency, the tail latency is left unaffected. This might be

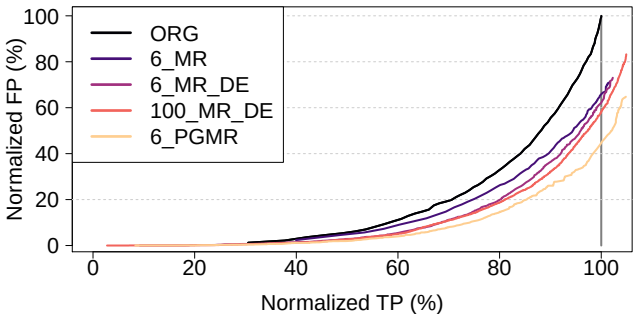


Figure 13: System configuration optimality analysis.

problematic on real-time applications that have a latency budget per input. As an important example, self-driving car systems are required to have a tail latency threshold of $100ms$ [42]. But, considering the overall low latency of baseline networks, it is still possible to satisfy the latency requirements while providing predictions with higher reliability. For example, ResNet34 as the most demanding network in our benchmark suite, requires less than $17ms$ on TITAN X (Pascal) to do a forward pass on single input.

Optimized 4_PGMR is also evaluated on a hardware with two GPUs similar to NVIDIA DRIVE AGX platform. In this scenario, CNNs are activated in a batch of two over available GPUs. As shown in Figure 12, the majority of inputs only require two networks to get the expected reliability. Therefore as shown in Figure 10b, the average latency can be reduced to baseline levels.

Discussion: To give perspective on the significance of these results, we consider an example of reliability improvement achieved by using networks with higher accuracy and complexity. We compare accuracy and cost of ResNet20 and DenseNet40 on the CIFAR10. Based on Table II, DenseNet40 reduces FP rate of ResNet20 by 18%. Whereas, MAC operations are increased from 41 MFLOPs to 267 MFLOPs, more than $6\times$ extra computation. This shows the inevitable trade-off between reliability and cost. In comparison, 4_PGMR on ResNet20 reduces FPs by 49.0% with $4\times$ cost, or by 46.3% with $1.6\times$ cost after optimizations. We observe that in this case, 4_PGMR is more cost effective to get higher reliability. It is good to mention that we are not suggesting to use PolygraphMR as a replacement for more accurate networks. Instead, our preference is to deploy PolygraphMR on top of them to achieve even a higher reliability as discussed in Section IV-B.

D. Preprocessing and Decision Engine

To show the impact of preprocessors and decision engine used in PolygraphMR, two separate experiments are run on CIFAR10 dataset with ConvNet. First, 6_PGMR is compared to a modified version of traditional MR, which deploys the smart decision policy proposed in Section III-E. We call this new system 6_MR_DE. By analyzing the changes in FP rates by moving from 6_MR to 6_MR_DE and from 6_MR_DE to 6_PGMR, we can observe the effect of decision engine and preprocessing, individually. Next, the 6_PGMR system is challenged by an extension of the modified MR used in the first experiment, which is assembled by training 100 copies of the baseline ConvNet.

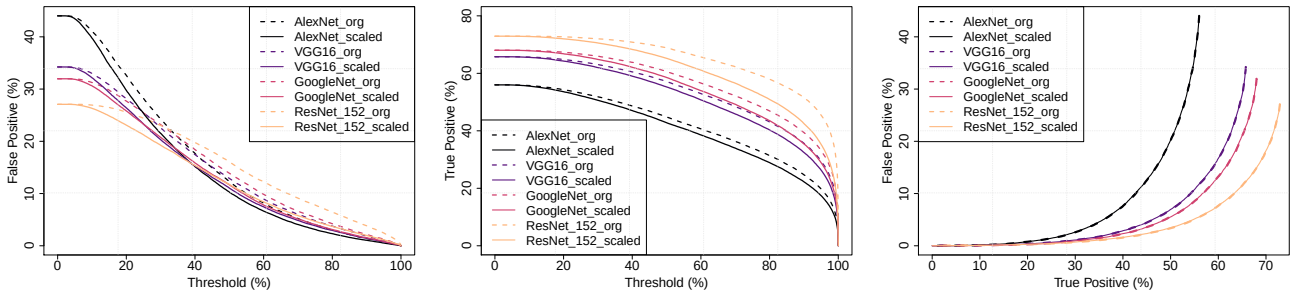
Figure 13 compares the Pareto frontier of different designs. To get the Pareto frontier of baseline CNN and 6_MR, they are coupled with a confidence threshold(Thr_Conf). It can be seen that both modified MRs are outperformed by PolygraphMR system. By comparing 6_PGMR with 6_MR_DE, we can see the extra gain of 18.5% in robustness introduced by preprocessing. We can also observe the gains from decision engine by comparing 6_MR_DE and 6_MR. The former provides 4.1% more normalized FP detection which is the result of using a smarter decision engine instead of just taking majority vote.

As for comparison of 6_PGMR and 100_MR_DE, even though the number of networks used in modified MR is 16 times more compared to 6_PGMR, the diversity introduced by using only 5 preprocessors is still higher. As a result, the reduction that 6_PGMR offers in the normalized FP rate is still 15.3% more than what 100_MR_DE can do.

E. Comparison with Network Calibration

Finally, we analyze network calibration as a solution to unreliability of confidence metric [17], [43]–[45]. As an experiment and to see the effects of calibration on the high confidence wrong answers, we implement the temperature scaling method proposed in the recent works [16]. Temperature scaling uses a scalar parameter to scale the output of softmax layer. The desired value of scaling for each benchmark is derived by solving an optimization problem [16].

The temperature scaling method is implemented and tested on 4 benchmarks over ImageNet dataset. Figure 14 shows the result prior to and after the temperature scaling. The dashed lines are for the original CNN and the solid lines report the results for scaled CNNs. The effect of confidence as a reliability metric is studied through Figure 14. Both Figure 14a and 14b show the effect of temperature scaling on FP or TP rate, with respect to the selected confidence threshold. If we compare FP and TP rate of the scaled and original network for each threshold value, we can see a reduction in both parameters. This gives the intuition that confidence of the undesired overconfident predictions is decreased which would make the confidence metric an appropriate reliability metric. But based on Figure 14c, we can see that Pareto frontier of TPs and FPs is unchanged after scaling. In other words, since the scaling is done by using the same parameter for all confidence values regardless of the correctness of answers, the final Pareto chart of the TPs and FPs is kept untouched. The only change is that for accessing a specific set of TP and FP, a lower confidence threshold is required for the scaled network. Hence, the initial reliability problem of confidence is still in place.



(a) False Positive (b) True Positive (c) Pareto Frontier of Confidence Threshold

Figure 14: Temperature scaling results

V. Related Work

We focus on different areas which target the reliability or security aspects of CNNs, or target to better understand the irrational behavior of CNNs.

Model uncertainty and Bayesian NNs are a closely related area of research [27], [46]–[48]. The idea is to add uncertainty to the predictions of NN models, and to be able to understand when the network is not confident with the generated results. Although the concept of model uncertainty is highly promising for reliable CNN inferences, but current solutions mainly target regression tasks [46], [47], or add a very high execution overhead, e.g., $10\times$ to $100\times$ in solutions based on the ensembles [27] or dropout sampling [46], [48]. However in PolygraphMR, we target improving classification reliability in CNNs, while considering the performance overhead implications of the proposed solution.

Researchers are also focusing on corner-case behaviors of the CNNs. They try to come up with systematic approaches and testing tools to detect erroneous behaviors [49], [50]. A number of works are also attempting to detect the anomalies in CNN applications. Their goal is to make CNNs more resilient against out-of-distribution examples that have not been seen before [15], [26], [51]–[53].

Security researchers are also focusing on the robustness of CNNs. They seek to find new techniques of generating adversarial inputs, to fool the network and induce their desired results. They add perturbations to the inputs, which in some cases is not even visible to the naked eye, leading to mispredictions [54]–[56]]. In contrast, they try to make the network robust against the state of the art adversarial generation methods [23], [57], [58].

A number of works are also focusing on the understandability and interpretability of CNNs in order to add more transparency to these algorithms. Samek et al. [59] try to explain the predictions of CNNs by measuring the sensitivity of the outputs to the individual input variables. Turner [60] proposes a general model for explaining the output of the classifiers. Zeiler et al. [61] propose a visualization technique to get an insight into the functionality of the intermediate features and operations of the classifier.

There are also numerous works on the reliability of CNNs against the transient faults and soft errors [62]–[64]. These works study fault injection in the CNN executions and analyze the response of the network to the faults. The solutions proposed for these reliability issues include modular redundancy in the level of application, instruction, and transistors. Our work however, focuses on the internal reliability problems of the DNNs which are also very vital considering that unlike transient faults, DNN prediction faults are quite common and happen frequently regardless of the hardware system that they are executed upon.

To the best of our knowledge, reliability problem of deep learning algorithms is a new topic for research in this area. Considering the increasing utilization of these algorithms in real world and vital applications such as autonomous vehicles, assuring their reliability needs to be well studied.

VI. Conclusion

CNNs are extensively utilized in mission critical applications such as autonomous vehicles. These applications require high levels of robustness since any error can cause irreparable damages. In this work, we demonstrated that current CNNs are failing to meet the reliability requirements of such applications by exposing a significant number of high confident mispredictions. We find that recent solutions that utilize confidence values as a metric of reliability are faulty and can lead to a significant number of false positives. We propose a new system called PolygraphMR composed of a number of CNNs as building blocks. Each network is accompanied by a specific preprocessor to provide behavior diversity among the CNNs and detect the unreliable wrong answers based on the contradictions observed due to the behavior variations. An energy efficient version of the system is also proposed that deploys precision reduction and staged activation to reduce the multiplicative costs of MR. As a result, our solution is capable to provide an average of 33.5% reduction in false positives of the original CNN, while requiring less than $2\times$ performance overhead.

Acknowledgment

We would like to thank our shepherd, Prof. Karthik Pattabiraman, and the anonymous reviewers for their comments and feedbacks. This work was supported by the National Science Foundation Grant NSF-XPS-1628991 and the Office of Naval Research Grant N00014-18-1-2020.

REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[2] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[3] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[4] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[5] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 3304–3308.

[6] J.-r. Xue, D. Wang, S.-y. Du, D.-x. Cui, Y. Huang, and N.-n. Zheng, "A vision-centered multi-sensor fusing approach to self-localization and obstacle perception for robotic cars," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 122–138, 2017.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[8] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[9] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, p. e4568, 2018.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

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

[13] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, 2017.

[14] M. He, S. Zhang, H. Mao, and L. Jin, "Recognition confidence analysis of handwritten chinese character with cnn," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 61–65.

[15] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[16] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*, 2017, pp. 1321–1330.

[17] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 625–632.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions." Cvpr, 2015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[22] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[23] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 582–597.

[24] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on*. IEEE, 2012, pp. 3642–3649.

[25] B.-K. Kim, H. Lee, J. Roh, and S.-Y. Lee, "Hierarchical committee of deep cnns with exponentially-weighted decision fusion for static facial expression recognition," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 2015, pp. 427–434.

[26] W. Wu, H. Xu, S. Zhong, M. R. Lyu, and I. King, "Deep validation: Toward detecting real-world corner cases for deep neural networks," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 125–137.

[27] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6405–6416.

[28] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[29] D. Siewiorek and R. Swarz, *Reliable Computer Systems: Design and Evaluatuion*. Digital Press, 2017.

[30] W. Bartlett and L. Spainhower, "Commercial fault tolerance: A tale of two systems," *IEEE Transactions on dependable and secure computing*, vol. 1, no. 1, pp. 87–96, 2004.

[31] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, N. E. Jerger, and A. Moshovos, "Proteus: Exploiting numerical precision variability in deep neural networks," in *Proceedings of the 2016 International Conference on Supercomputing*. ACM, 2016, p. 23.

[32] P. Hill, A. Jain, M. Hill, B. Zamirai, C.-H. Hsu, M. A. Laurenzano, S. Mahlke, L. Tang, and J. Mars, "Deftnn: Addressing bottlenecks for dnn execution on gpus via synapse vector elimination and near-compute data fission," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 786–799.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[35] A. Krizhevsky, "cuda-convnet: High-performance c++/cuda implementation of convolutional neural networks," 2012.

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[38] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *2009 IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, 2009, pp. 163–174.

[39] J. Leng, T. Hetherington, A. ElTantawy, S. Gilani, N. S. Kim, T. M. Aamodt, and V. J. Reddi, "Gpuwattch: enabling energy optimizations in gpgpus," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3, pp. 487–498, 2013.

[40] M. Khairy, J. Akshay, T. Aamodt, and T. G. Rogers, "Exploring modern gpu memory system design challenges through accurate modeling," *arXiv preprint arXiv:1810.07269*, 2018.

[41] NVIDIA, "Nvidia drive agx self driving compute platform," 2011, https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/.

[42] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2018, pp. 751–766.

[43] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *ICML*, vol. 1. Citeseer, 2001, pp. 609–616.

[44] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning." in *AAAI*, 2015, pp. 2901–2907.

[45] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[46] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[47] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[48] S. Khan, M. Hayat, S. W. Zamir, J. Shen, and L. Shao, "Striking the right balance with uncertainty," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 103–112.

[49] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 303–314.

[50] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 1–18.

[51] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *arXiv preprint arXiv:1706.02690*, 2017.

[52] T. DeVries and G. W. Taylor, "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[53] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.

[54] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[55] S. M. Moosavi Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. EPFL-CONF-218057, 2016.

[56] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[57] J. Gao, B. Wang, Z. Lin, W. Xu, and Y. Qi, "Deepcloak: Masking deep neural network models for robustness against adversarial samples," 2017.

[58] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," *arXiv preprint arXiv:1703.04318*, 2017.

[59] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *arXiv preprint arXiv:1708.08296*, 2017.

[60] R. Turner, "A model explanation system," in *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 2016, pp. 1–6.

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

[62] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2017, p. 8.

[63] S. Bettola and V. Piuri, "High performance fault-tolerant digital neural networks," *IEEE transactions on computers*, vol. 47, no. 3, pp. 357–363, 1998.

[64] V. Piuri, "Analysis of fault tolerance in artificial neural networks," *Journal of Parallel and Distributed Computing*, vol. 61, no. 1, pp. 18–48, 2001.