Diagnostic mechanism and robustness of safety relevant automotive deep convolutional networks

Krutsch Robert
Intel
Munich, Germany
Robert.Krutsch@intel.com

Dr. Rolf Schlagenhaft
NXP
Munich, Germany
Rolf.Schlagenhaft@nxp.com

Abstract

Convolution Neural Networks today provide the best results for many image detection and image recognition problems. The computational complexity and the amount of parameters learned has increased, yet there is little to no research on the topic of functional safety for systems incorporating CNNs. The analysis on false detections due to random hardware faults concentrates on human made adversarial examples obtained by adding unrealistic noise sources over carefully selected images. Redundant execution of these networks is prohibitive in application domains where power and price constraints dominate, pushing for alternate approaches. In this paper we investigate functional safety aspects for a road labeling application, a common task in the advance driver assistance systems. We introduce computationally light safety checks that reduce the error space significantly, train a CNN on the Cityscape dataset that reaches 93% mean IU (intersection over union) and use Monte Carlo simulations to assess the impact of single event upset random hardware faults. The results show that the networks based on convolution and ReLU (rectified linear unit) have some intrinsic robustness and that together with additional constraints strong function safety claims can be made. We compare also the diagnostic coverage between floating point and fixed point implementation of CNNs and summarize key safety features needed to achieve a high diagnostic coverage.

1. Introduction

Convolutional neural networks have been driving major advances in whole image classification [1, 2, 3], semantic segmentation [4] and object detection [5, 6]. The advances in accuracy were achieved by dedicating more and more parameters [1, 2] or more sophisticated and deeper networks [3]. The growing amount of data and compute capability of the training infrastructure allows for growing the parameter space in the coming years and for better and better accuracy.

The hardware architectures used to deploy CNNs today are based on parallel computation engines that are limited mainly by memory bandwidth constraints. For example, the well-known AlexNet structure [1], for image input resolution of 227x227 pixels needs 721 Mega Multiply Accumulate Operations/frame and 224MB/frame data transfers (considering an ideal, one time read that does not really apply for all existing architecture e.g. GPUs [7]).

In embedded systems the computation and bandwidth resources are limited by the power consumption requirements that follow naturally from temperature profile of the overall system. The domain is also very competitive, pushing for efficient and cost effective implementations.

Functional safety aspects are addressed in a mix of software and hardware, yet target mainly the clock domain of the SoC, power domain, permanent errors through periodic checks and soft errors through parity and error correction codes. At the software level a multi-path algorithmic approach is taken, where machine learning algorithms are combined with optical flow and stereo disparity for a more reliable detection. Temporal consistency of the detections, redundancy in the key areas as well as a code tracing mechanisms insure that errors do not make their way to erroneous and dangerous automated driving actuations. These methods heavily rely on the driver being alert to detect eventual false negatives and are programmed to give control to the driver in case of failure. The fully automated driving architectures, where such a safe state is not easily assumable, need to do significantly better on the functional safety aspects to gain acceptance in the automotive space (especially now where almost as many cars are recalled as produced, e.g. 2015 and 2016).

In this paper we introduce a road labeling convolutional neural network trained and tested on the Cityscapes dataset that has lower computational need but still manages to keep a high mean Intersection over Union (IU) value. We introduce a mathematical framework of error detection giving a diagnostic coverage for a certain part of the space of the effects of random hardware faults. For the remaining uncovered space of fault effects we perform a Monte Carlo simulation to assess its dangerousness. Generic error injection is very hard to implement because of the significant amount of possible sources; our “safety checks” allow us to limit this space and test only in a limited subspace.

The paper is structured as follows: Chapter 2 discusses some of the prior art, in Chapter 3 the road labeling network is introduced while chapter 4 discusses the functional safety related topics.

2. Related Work

The image segmentation task is a well-known and very often addressed problem in the field of advanced driver assistance. In 2016 the Cityscapes dataset was introduced [11] and many publication and approaches, existing and new, were exercised based on this database. State of the art road labeling schemes achieve over 95% mean IU but often compromise on execution speed to achieve better and better detection performances (e.g. uses VGG [2] , GoogleNet or AlexNet [1]). We employ a simplified network with only 100000 parameters, dedicated to detecting the road. The results can be used to generate proposal areas for other classifiers and reduce the search space and computational demands for following detectors that need to find small structures in the image. The need to detect smaller objects is understandable considering a simple example where for a camera resolution of 640x480 , focal length 7.5mm and FoV(Field of View) of 27%, a 30pixel pedestrian is just 4s away (considering a speed of 55km/h , [13]). Although the image height has more than doubled in the recent years and the FoV of the cameras has also increased to cope with use cases when vulnerable road users enter the drivable path from the sides the problem of small structure detection is still of interest.

In the field of functional safety, the ISO26262 [14] gives guidelines and requirements at the system level, defining the framework on how hardware and software is built. The framework does not mandate specific implementation features, so the designer has the freedom and the responsibility of how to achieve functional safety for the final product. In the domain of semiconductors, the following main fault models have to be considered: permanent faults that do not vanish (that can have as cause for e.g. equipment wearout), transient faults that are unexpected changes due to the physical media, which are typically short lived. The permanent errors are typically detected through hardware mechanisms (e.g. logic built-in self-test) that are beyond the scope of this exposition; we concentrate on the transient errors since these are more challenging to detect and handle.

The CNN structures are typically run on massive parallel hardware accelerators that have tightly coupled memories (Fig 1). Today's state of the art implementations have parity or error correction on the entire path from the DDR memory to the tightly coupled memory eliminating a significant amount of possible errors. The errors on the logic side are either argued away because they might affect single pixels (e.g. errors in simple image filtering) or excluded through redundancy or through higher level intelligence about the overall system. The logic part of the accelerator is always a gray area where the engineering judgment is used to find smart diagnostic mechanisms that can lead to high diagnostic coverages without complete redundancy. This work proposes such a scheme for the convolutional layer.

In [10] and [12] adversarial examples that lead to misclassification are produced, yet the papers considers possible errors that are not seen in practice and take into account only errors in the input image, ignoring intermediate layer results. The papers also employ mining for low confidence classifications to find adversarial examples more easily.

In [15] a Monte-Carlo testing approach in a simulated environment is proposed. The tests are conducted at system level without decomposition into building blocks. Due to the large error spaces of individual blocks and interaction of elements such methods are impractical for modern architectures. Our paper concentrates on CNN implementation only and combines mitigation mechanisms with Monte-Carlo testing to limit the simulation space.

Another approach taken by the industry is based on "Brute Force", multiple cars drive millions of miles with the system under test. While this methods are necessary for algorithmic development they are not statistically relevant from functional safety perspective considering the error rates and possible dangerous error space. An increase in significance could be achieved if error injection is performed and the system is compared to the error free prior, yet even so the overall system complexity would allow only limited claims on system coverage.

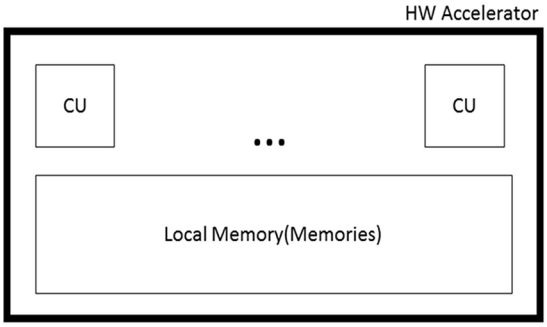


Figure 1: Hardware accelerator – abstract block diagram

3. Road Labeling

To be able to run the Monte Carlo simulations quicker and to be able to deploy the network on low-end embedded hardware we have decided to investigate lower complexity networks for the task of the road labeling. For the training process we have used the Cityscapes database [11] but have decimated the image size by a factor of two in each dimension. During the training process we have experimented with various networks ranging from 6 giga

multiply accumulate for 30 frames per second up to 35 giga multiply accumulate; all networks achieved over 90% mean IU on the test dataset of Cityscapes for the road class (note, our network is used only to discriminate road from non-road and does not discern between any other classes). We have decided to use for this investigation the network presented in Table 1 since it is a good compromise of speed and accuracy. The mean IU obtained on the down-sampled version of Cityscape is around 93%. We have trained the network with a learning rate of 10e-9 for 20000 iterations and decreased it to 10e-10 afterwards while keeping the deconvolution layers with constant bilinear up-sampling coefficients. Learning the deconvolution coefficients did not bring a significant gain in accuracy. Note that the network uses 3 up-sampling layers with smaller kernels and not one layer with a bigger kernel, the reason behind is embedded hardware implementation constraints where excessive big kernels can lead to inefficiencies. Some of the results of the network can be visualized in Figure 2.

Figure 2: Output of the road labeling network. With green the detected road is marked in the image.

Layer	Kernel Size	Stride	Channels
Image	-	-	3
Conv1+Relu	5x5	2	32
Conv2+Relu	5x5	2	32
Conv3+Relu	5x5	2	32
Conv4+Relu	5x5	1	48
Conv5+Relu	5x5	1	2
Deconv1	4x4	2	2
Deconv2	4x4	2	2
Deconv3	4x4	2	2

Table 1: CNN network for road labeling

4. Functional safety aspects

4.1. Discussion on error sources

Considering the typical hardware architecture of vision accelerators and the task of CNN forward implementation we identify the following main error categories:

- Error in the code and control flow
  - Errors in the code
  - Errors in state machines and hardware pipelines during execution
- Error in data processing

The errors in the code and control flow can be reasonably well detected by protecting the memories and by augmenting the normal data with test patterns. At the end of the processing, consistency checks can be done by another hardware engine, for e.g. checking a few of the outputs, checking that the amount of lines that were supposed to be processed were also processed. Detection of illegal instruction, watchdog monitoring and proper handling of the interconnect protocol at the SoC level are also common mechanisms.

The processing faults are more challenging to detect, contain and potentially correct. The errors that might appear in memories can be detected and eventually corrected when using error correction mechanisms while the errors in the processing hardware (e.g. registers, multipliers, adders) are mainly addressed through redundancy or are ignored (for certain algorithms a single pixel error does not have any impact). Another approach, that is applicable only for certain classes of algorithms, is based on consistency checks having in mind possible bounds of the computation performed. We present such an approach for convolutional layers, by employing a scheme that does not use any multiplication. The lack of multiplications is essential for embedded hardware due to the significant size of the multiplier compared to comparators and adders (a multiplier is more than 20 times bigger than an adder).

When analyzing possible fault sources for convolution layers we find the following possible fault categories:

- errors in weights
- errors in feature maps or input data
- errors in intermediate accumulators

While the most frequent transient faults are single and dual bit faults it is very difficult to limit the space of resulting errors after the computation is done; dependent on input data, network coefficients and hardware architecture the error might manifest itself in the output as any arbitrary number in the possible value range. One interesting aspect to note is exactly the way numbers are represented, for example the fixed point representation has equally spaced numbers while for floating point the numbers are concentrated around zero and the gap between numbers increases exponentially. This observation is especially interesting when computing the coverage of a certain test based on validity ranges, for example bounding an error between -0.5 and 0.5 for single precision floating point means that the coverage is only around 51% (computed as numbers outside the interval divided by all possible values). Such small singular errors might have little to no effect on CNNs, especially if the network is trained also on samples with synthetic noise added yet we investigate this topic through Monte Carlo simulations.

Another interesting observation on the robustness to errors of popular CNN architectures is the error masking behavior of the RELU function. The layer has the interesting property of masking errors if the input is negative and the error does not change the state of the sign bit. Thresholds over the output confidence have similar effect, yet harder to trace back to induvial neurons in the network.

It is also natural to expect that the impact on the overall result depends also on where the error manifests itself (if it is in one of the first layers or if it is in the last layers). Another interesting aspect is if the network exhibits some "attention" patterns; if errors that influence the results in a certain part of the image are propagated selectively (e.g. the road in the Cityscape dataset is not present everywhere in the image).

4.2. Error detection mechanism

The proposed detection mechanism is based on checking whether the processing result lies within a mathematically determined valid range. In the following the constraints are elaborated and their test coverage is investigated.

Lemma 1. Given a convolution function of form (1) there exists a real value *Q* such that (2) holds true (we have used the Einstein summation convention for compactness).

$$f(x) = w^i x_i + b, i = \overline{1,N}; f: R^N \to R \qquad (1)$$

$$|f(x) - f(x^0)| \le Q \sum_{i=1}^{N} |x_i - x_i^0| \qquad (2)$$

$(\mathrm{x}^0, \mathrm{f}(\mathrm{x}^0))$ – is called a seed, can be any tuple from the input output domain

Proof - The inequality is trivial to prove, and follows immediately from the triangle inequality.

$$|w^i x_i + b - w^i x_i^0 - b| \le \sum_{i=1}^{N} |w^i||x_i - x_i^0|$$

$$\le \max(|w^i|) \sum_{i=1}^{N} |x_i - x_i^0| = Q \sum_{i=1}^{N} |x_i - x_i^0| \qquad (3)$$

Where

$$Q \equiv \max(|w^i|)$$

Lemma 2. The interval where possible errors are not detected by (2) is bounded and the bounds depend on how the seed is selected, given an input space.

The lemma follows the intuitive question, what errors can be detected by the inequality; for what part of the error space does the inequality still hold?

We consider the error as an additive value over the true, uncorrupted, output:

$$f_{err}(x) = f(x) + e \qquad (4)$$

We transform the inequality into a function that depends on the error:

$$F(e) \equiv |f(x) + e - f(x^0)| - Q \sum_{i=1}^{N} |x_i - x_i^0| \qquad (5)$$

Lemma 2 can then be rephrased as finding *e* for that $\mathrm{F(e)} \le 0$. For simplicity we use the following notation:

$$A \equiv f(x) - f(x^0); B \equiv Q \sum_{i=1}^{N} |x_i - x_i^0| \qquad (6)$$

The inequality is trivial to solve, for space reasons we give here only the result:

$$e \in [B - A, B + A] \qquad (7)$$

As can be observed from (7), the interval where the inequality does not detect the potential error, is bounded and it can reduce the possible error space significantly. The size of the interval depends on A, the smaller this value is the tighter the inequality holds. This allows for constructing a lookup table with seeds that can minimize the error interval size once a mechanism for selecting the seed is found.

Lemma 3 Minimizing the cumulative error for the entire input space can be done for each input dimension separately and is achieved by halving the input range.

The size of the interval from (7) is: $D \equiv 2A$

For a one dimensional case the problem can be rewrite as $M = \min_{x^0} \int D\,dx$; in the multidimensional case the integral is multidimensional, depending on all the x_i .

For the simple one dimensional case, the problem is trivial and considering an interval of *[0, 1]* for the input values we can see that the optimal value for x^0 is 0.5 (follows from eq. (8)).

$$M = \min_{x^0}(2Q\int_0^{x^0}(x^0 - x)dx + 2Q\int_0^1 (x - x^0)dx)$$

$$M = \min_{x^0}((x^0)^2 - x^0 + 0.5) \qquad (8)$$

For the multidimensional case, given the fact that there are no cross dimension factors, the integral can be rewritten as a sum of single dimension integrals. Each integral is positive, being made out of absolute values, such that the minimum is achieved where the minimum of the individual elements is found ((9) for a normalized input space).

$$M = \min_{x^0} \int .. \int 2\mathrm{Q} \sum_{\mathrm{i}=1}^{\mathrm{N}} |\, x_i - x_{\mathrm{i}}^0 \,| dx_1 .. dx_N$$

$$M = 2Q \min_{x^0} \sum_{\mathrm{i}=1}^{\mathrm{N}} \int |\, x_i - x_{\mathrm{i}}^0 \,| \mathrm{dx_i} \qquad (9)$$

Note that for an un-normalized input space the individual integrals will enter with a factor, so placing the seed smartly

for this un-normalized spaces has a high impact in the overall tightness of the inequality, influencing the error detection capabilities.

Lemma 3 assumes a white noise like distribution of the data in the interval, each value has an equal probability of being found. In practice, this is not often the case and it is only natural if we think about the image space where the sky is typically in the upper part of the image, the road on the lower part, etc. The number representation and quantization introduces also this types of effects, making the space curved (e.g. floating point number have an exponential representation with more precision closer to zero and less once going to larger numbers).

4.3. Error coverage

One of the first questions that arises is the coverage of the tests given the possible error space. The coverage is computed as:

$$C = 1 - \frac{cardinality(D)}{number\_representation}[\%] \qquad (10)$$

Where D is the set where errors are not detected while the number representation is the total amount of numbers that can be represented (depends on the bit-depth and representation scheme). We have computed the average value of *C* for the first convolution layer considering the following assumptions (Table 2):

- [Case1] Floating point numbers representation; the seed value is set in the middle of the value ranges observed for the convolution maps
- [Case2] Floating point numbers representation; the seed value is set on the average value of the convolution map
- [Case3] Fixed point number representation (Q15 representation for inputs and weights and Q7.24 for the output feature maps); the seed value is set on the average value of the convolution maps. The number representation was chosen based on following considerations : weights are between -1 and 1 (Q15 gives the best fractional fidelity for this range) and filter sizes are 5x5x3 demanding at least 7 bit integer part for guaranteed overflow less operations.

Case 1	Case 2	Case 3
45%	47%	97%

Table 2: Average coverage for convolution 1

As can be observed the average coverage for the floating point representation is very low (considering desired diagnostic coverage of over 90%) due to the fact that the numbers are concentrated around zero and our bounding interval contains this region very often. The floating point representation , due to the exponential form, pushes the optimal seed value towards zero while for fixed point the selection of the seeds is more natural and is done based on the spikes of the distribution of occurrence of values. The coverage for floating point could be improved if we could show that the algorithm is robust to small errors; for e.g. if numbers in the range [-0.5, 0.5] have low impact in the output. To test this supposition we inject errors from this interval and asses the output of the network. For the fixed point representation minimum coverage observed was around 77%. The coverage tests are based on ~1.6 billion samples, generated with the test set of the Cityscapes dataset.

4.4. Error injection

We have injected over 500 million errors over all possible positions in the feature maps considering the following test cases:

- [TC1] errors are injected in the interval where the test does not offer any coverage, considering a placement of the seed as in [Case 1]
- [TC2] errors are injected in the interval [-0.5,05]

Only one single error at a time is injected and the final output of the network is compared with the output without error injection. Errors are generated based on a normal distribution random number generator. The implementation is done based on single precision floating point number representation and the input statistics of the images can be visualized in the histograms in Figure 3.

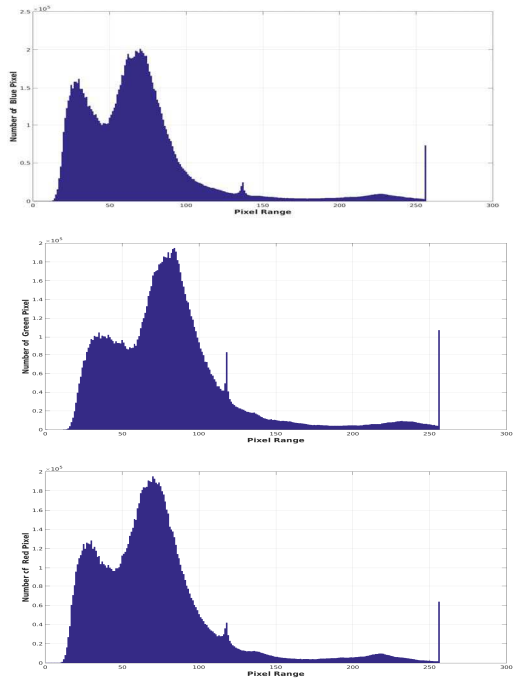


Figure 3: Cumulated histograms over all input images used in the Monte Carlo simulation.

We consider that a 1-2% degradation in detection will minimally impact the next processing elements that estimate

the road model and generate the region of interest for the object classifiers. This assumption is generally accepted for RANSAC (random sample consensus) model estimation where even a higher amount of outliers could be rejected.

We are interested to see the number of errors and the position where the errors were injected in the feature map. To better visualize the results we construct a grid of the same size as the feature map; in each cell of the grid we cumulate the errors in the output when injecting noise in that location in the feature maps (e.g. for cell position (100,100) corresponding to the first convolution layer we cumulate the output errors coming from injecting errors in feature map positions (100, 100, 1:32), where 32 represents the number of output feature map of that layer). The results, for the two test cases can be visualized in Figure 4; the grids are normalized with their highest values to bring them in the image visual space. An interesting aspect of the results is the fact that the network seems to be less affected by errors injected in the upper part of the image (Figure 4a), suggesting that it learned to suppress this area more heavily than others. This is intuitively consistent with the fact that the road is placed in the lower part of the images as can be observed in Figure 5 and that almost in all images the sky has a similar texture. In Figure 4, stripe patterns can be observed that follow the dimension of the kernel and stride used in the network; the errors are spread into the neighbors by the convolution filters, according to the receptive field size.

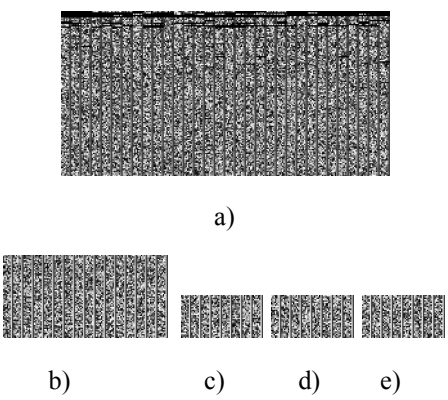
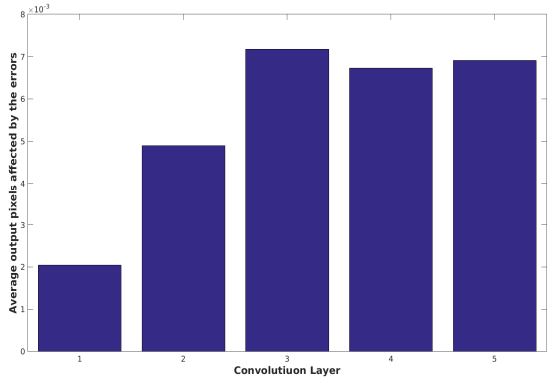


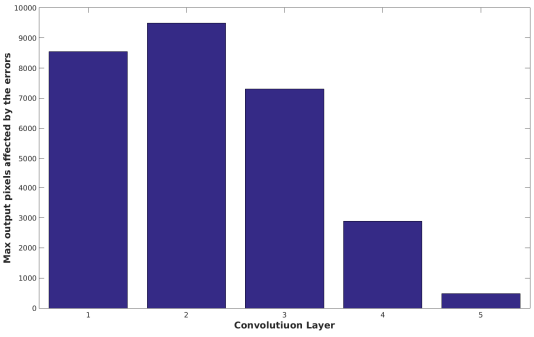
Figure 4: Error grids, lighter gray values represent more errors manifested in the output when injecting errors in that location a) First convolution layer, b) Second convolution layer, c) 3rd convolution layer, d) 4th convolution layer, e) 5th convolution layer. The results were obtained by injecting over half a billion errors, spread over all possible elements of the feature map.

Figure 5: a) Cumulated area in the images of the Cityscapes dataset where the road is present; marked with gray color in image above

Another aspect that we would like to understand is if the errors in the first convolution layers have a higher or lower impact than in later convolution layers. We construct histograms with the cumulate errors obtained from the different layers and normalize them by the number of test we have performed and also record the maximum amount of changes in the output label mask due to an error (Figure 6). As can be observed, the errors in the first layers have less average impact (Figure 6a) yet can have high individual impact as can be observed in Figure 6b where more output pixel labels have been changed due to errors injected in the first layers.



a)

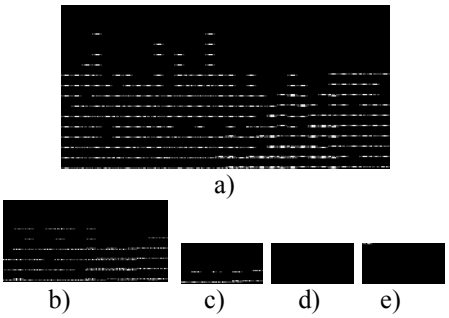


b)

Figure 6: a) Average output errors for each of the five convolution layers; b) Maximum amount of pixels in the output mask affected by an error.

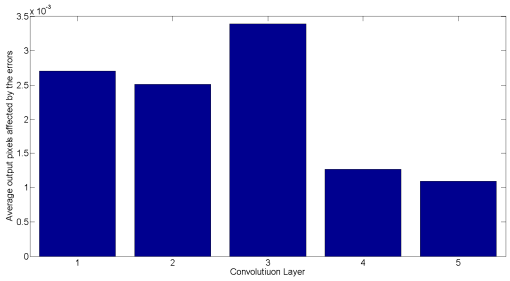
Note that the amount of errors introduced can give only a hint on the overall influence of soft errors. As can be observed in Figure 4 the effect of error injection is uniform and one would expect this if the data set and injection patterns would have reasonable size, yet they do not cover all the possible error space (for a -0.5, 0.5 range and a single image, considering only the first convolution layer we would need to inject over 22 trillion errors to cover only 1% of the error space). What can be also observed is that the fixed point variant has a better out of the box coverage than the floating point counterpart due to the compression of the dynamic range around zero.

For TC2 we have injected noise between [-0.5, 0.5] from a uniform distribution. Similar to TC1 we have aggregated the data in form of grids (Figure 7) and also computed the average pixels selected and the maximum pixel affected by the error injection (Figure 8). We can observe that the average and maximum values decrease compared to TC1 yet the amount of pixels affected is significant (more than 3%) in certain cases and needs some careful assessment if it is still usable for the next processing steps.

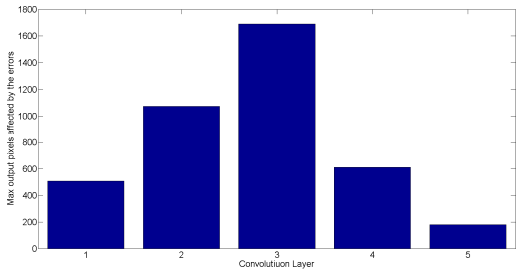


a) b) c) d) e)

Figure 7: Error grids, lighter colors represent more errors manifested in the output when injecting errors in that location a) First convolution layer, b) Second convolution layer, c) 3rd convolution layer, d) 4th convolution layer, e) 5th convolution layer. The results were obtained by injecting over half a billion errors, spread over all possible elements of the feature map.



a)



b)

Figure 8: a) Average output errors for each of the five convolution layers; b) Maximum amount of pixels in the output mask affected by an error.

5. Conclusions and future work

In this paper we have introduce a computationally light network for road segmentation and studied generic approaches for functional safety tailored to CNN architectures. We have introduced a mitigation technique that has low computational requirements and that gives good coverage numbers for fixed point implementations. For a larger deployment and test the infrastructure of error injection needs to be improved. We have used the Caffe framework that could be extended with error injection layers that would restrict the computation only to the feature map points affected by an error (only in the receptive affected); in our implementation we run the complete computation.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[2] Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. In CoRR, abs/1512.03385, 2015.

[4] Jonathan Long et al., Fully Convolutional Networks for Semantic Segmentation. In CoRR, abs/1411.4038, 2014.

[5] Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In ICLR, 2014

[6] Ross B. Girshick et al,. Rich feature hierarchies for accurate object detection and semantic segmentation. In CoRR, abs/1311.2524, 2014.

[7] Robert Hochberg, Matrix Multiplication with CUDA - A basic introduction to the CUDA programming model, 2012

[8] GPU - Based Deep Learning Inference: A Performance and Power Analysis, Whitepaper, November 2015.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan,

[10] I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013. 1, 5, 7, 8

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016

[12] Anh Nguyen et. all .Deep Neural Networks are Easily Fooled: High Confidence Prediction for Unrecognizable Images. arXiV preprint arXiV:1412.1897v4, 2015

[13] Piotr Dollar et all, Pedestrian Detection : A Benchmark . CVPR 2009

[14] ISO26262 – Road Vehicle Functional Safety

[15] Daniel Meltz , Hugo Guterman Verification of Safety for Autonomous Unmanned Ground Vechicles, 2014 IEEE 28-th Convention of Electrical and Electronics Engineers Israel