MANCHESTER 1824

The University of Manchester

QNNRepair: Quantized Neural Network Repair

Xidan Song[1], Youcheng Sun[1], Mustafa A. Mustafa[1] [2], Lucas C. Cordeiro[1]

[1]Department of Computer Science, University of Manchester, UK

[2]imec-COSIC, KU Leuven, Belgium

MANCHESTER
1824
The University of Manchester

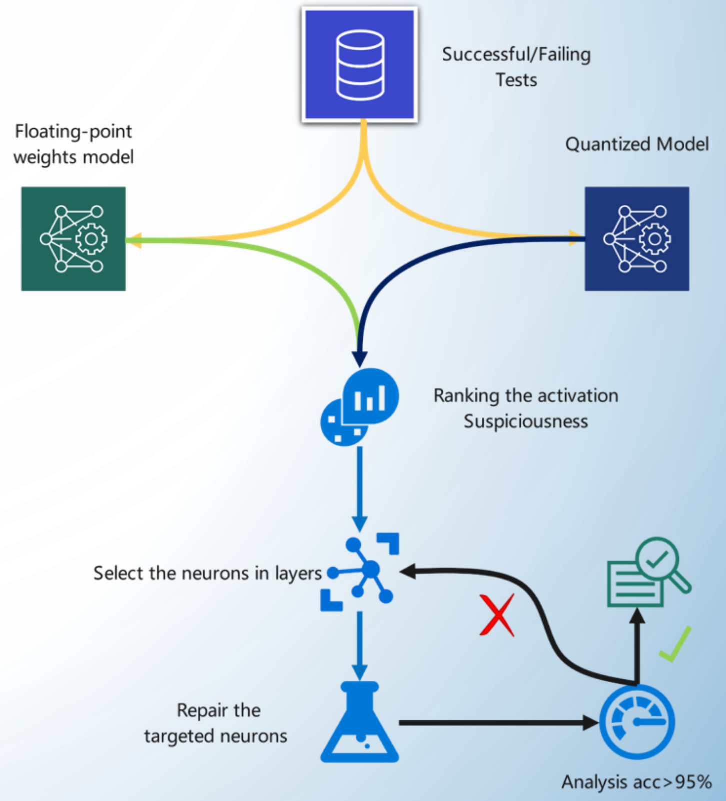
QNNRepair: Quantized Neural Network Repair

- QNNRepair is a new method for repairing QNNs.
- It converts quantized neural network repair into a MILP (Mixed Integer Linear Programming) problem.
- We compare QNNRepair with a state-of-the-art QNN repair method–Squant, and demonstrate that QNNRepair can achieve higher accuracy than Squant after repair.
- We also evaluate QNNRepair on multiple widely used neural network architectures to demonstrate its effectiveness.

QNNRepair Architecture

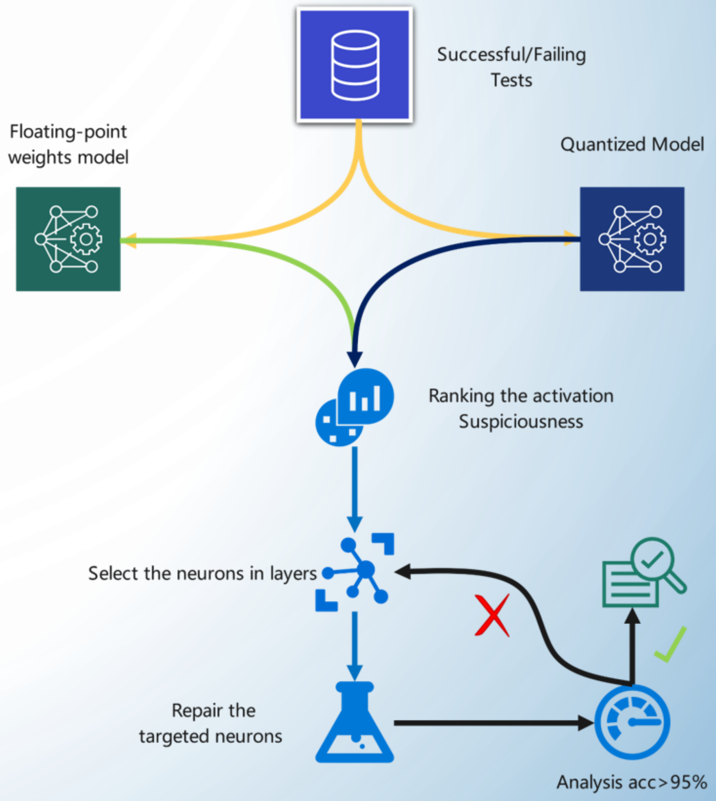
Input data:

- Repair datasets of successful (passing) and failing tests, the two models would produce the same classification outcome when given the same test input.
- Used by QNNRepair to evaluate each neuron’s importance and localize these neurons to repair.
- Can be generated by dataset augmentation or various neural network testing methods


Successful/Failing Tests
Floating-point weights model
Quantized Model
Ranking the activation Suspiciousness
Select the neurons in layers
Repair the targeted neurons
Analysis acc>95%

QNNRepair Architecture

Ranking Neurons:
Evaluating the importance of the neurons in the neural network for causing the output difference between the quantized model and the floating point one.


Successful/Failing Tests
Floating-point weights model
Quantized Model
Ranking the activation Suspiciousness
Select the neurons in layers
Repair the targeted neurons
Analysis acc>95%

Ranking Neurons

$$\begin{pmatrix} f_{11} & \cdots & f_{1n} \\ \vdots & \ddots & \vdots \\ f_{m1} & \cdots & f_{mn} \end{pmatrix} = f_i \text{ and } q_i \text{ for the quantized model.}$$

Given an input image i, we calculate $diff_i = f_i - q_i$ between the two models. We form a large matrix of these $diff_i$ regarding the image i.

Ranking Neurons

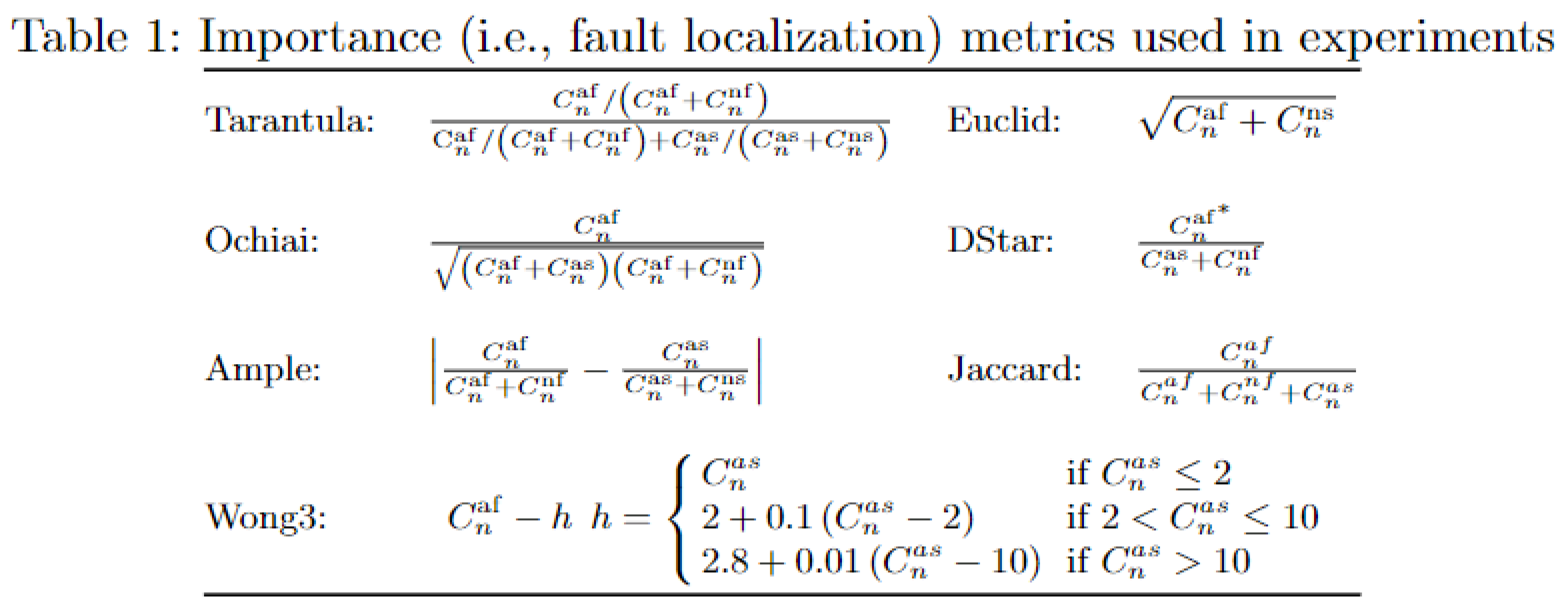


Table 1: Importance (i.e., fault localization) metrics used in experiments

Tarantula:	$\frac{C_n^{\mathrm{af}}/(C_n^{\mathrm{af}}+C_n^{\mathrm{nf}})}{C_n^{\mathrm{af}}/(C_n^{\mathrm{af}}+C_n^{\mathrm{nf}})+C_n^{\mathrm{as}}/(C_n^{\mathrm{as}}+C_n^{\mathrm{ns}})}$	Euclid:	$\sqrt{C_n^{\mathrm{af}}+C_n^{\mathrm{ns}}}$
Ochiai:	$\frac{C_n^{\mathrm{af}}}{\sqrt{(C_n^{\mathrm{af}}+C_n^{\mathrm{as}})(C_n^{\mathrm{af}}+C_n^{\mathrm{nf}})}}$	DStar:	$\frac{C_n^{\mathrm{af}^*}}{C_n^{\mathrm{as}}+C_n^{\mathrm{nf}}}$
Ample:	$\left\lvert\frac{C_n^{\mathrm{af}}}{C_n^{\mathrm{af}}+C_n^{\mathrm{nf}}}-\frac{C_n^{\mathrm{as}}}{C_n^{\mathrm{as}}+C_n^{\mathrm{ns}}}\right\rvert$	Jaccard:	$\frac{C_n^{af}}{C_n^{af}+C_n^{nf}+C_n^{as}}$
Wong3:	$C_n^{\mathrm{af}}-h \;\; h=\begin{cases} C_n^{as} & \text{if } C_n^{as}\leq 2 \\ 2+0.1\,(C_n^{as}-2) & \text{if } 2<C_n^{as}\leq 10 \\ 2.8+0.01\,(C_n^{as}-10) & \text{if } C_n^{as}>10 \end{cases}$		

– C_n^{af} is the number of “activated” neurons for failing tests.
– C_n^{nf} is the number of “not activated” neurons for failing tests.

– C_n^{as} is the number of “activated” neurons for passing tests.
– C_n^{ns} is the number of “not activated” neurons for passing tests.

The University of Manchester

Ranking Neurons

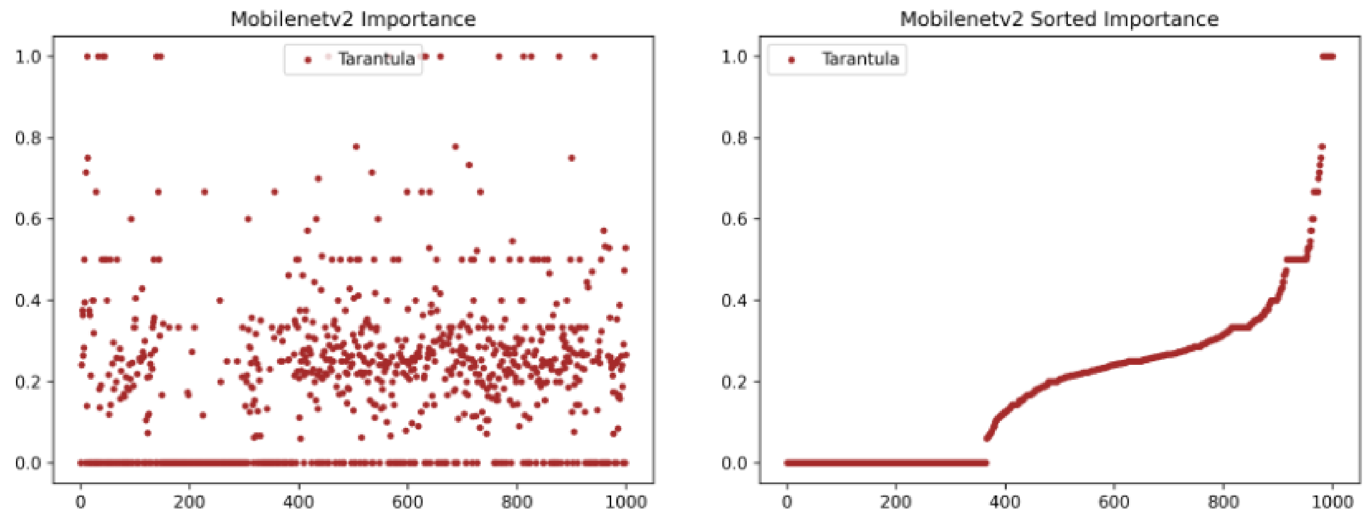


Fig. 2: Importance distribution regarding certain importance metrics on MobileNetV2.

Ranking Neurons

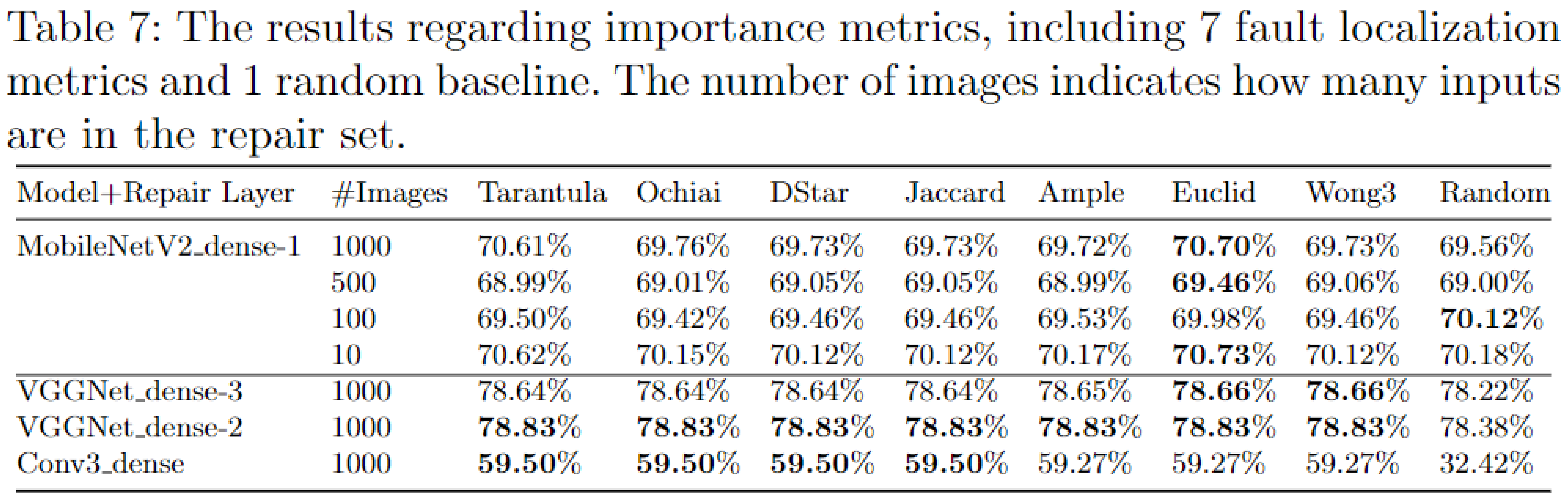


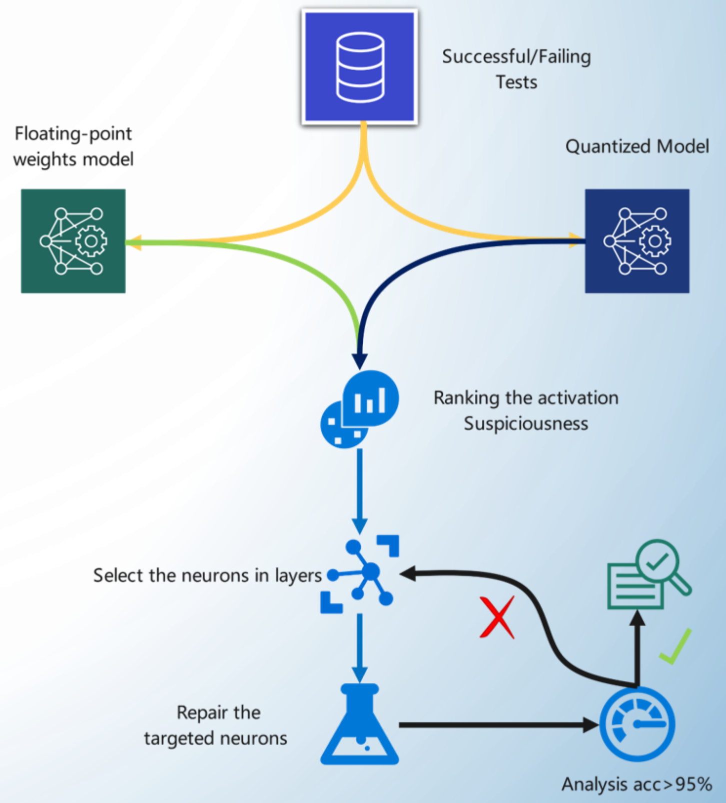
Table 7: The results regarding importance metrics, including 7 fault localization metrics and 1 random baseline. The number of images indicates how many inputs are in the repair set.

Model+Repair Layer	#Images	Tarantula	Ochiai	DStar	Jaccard	Ample	Euclid	Wong3	Random
MobileNetV2_dense-1	1000	70.61%	69.76%	69.73%	69.73%	69.72%	**70.70%**	69.73%	69.56%
	500	68.99%	69.01%	69.05%	69.05%	68.99%	**69.46%**	69.06%	69.00%
	100	69.50%	69.42%	69.46%	69.46%	69.53%	69.98%	69.46%	**70.12%**
	10	70.62%	70.15%	70.12%	70.12%	70.17%	**70.73%**	70.12%	70.18%
VGGNet_dense-3	1000	78.64%	78.64%	78.64%	78.64%	78.65%	**78.66%**	**78.66%**	78.22%
VGGNet_dense-2	1000	**78.83%**	**78.83%**	**78.83%**	**78.83%**	**78.83%**	**78.83%**	**78.83%**	78.38%
Conv3_dense	1000	**59.50%**	**59.50%**	**59.50%**	**59.50%**	59.27%	59.27%	59.27%	32.42%

QNNRepair Architecture

Repair The Targeted Neurons:

- After the neuron importance evaluation, for each layer, we obtain a vector of neuron importance. We rank this importance vector.
- The neuron with the highest importance is our target for repair as it could have the greatest impact on the corrected error outcome.


Successful/Failing Tests
Floating-point weights model
Quantized Model
Ranking the activation Suspiciousness
Select the neurons in layers
Repair the targeted neurons
Analysis acc>95%

MANCHESTER 1824

The University of Manchester

Repair the targeted Neurons

Minimize: M

Subject to:

$M \geq 0$

$\delta_i \in [-M, M] \quad \forall i \in \{1, 2, \ldots, n\}$

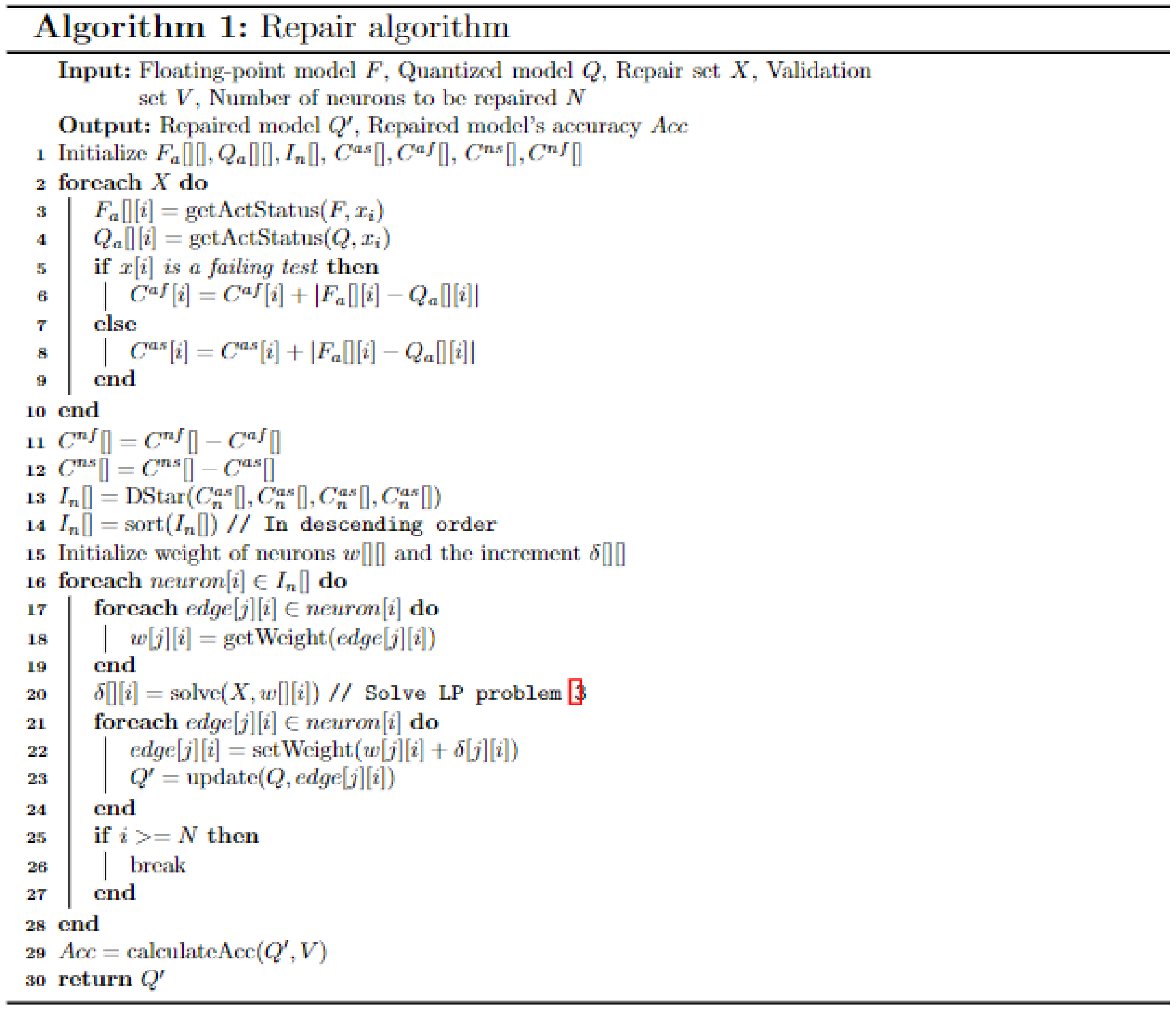
If floating model gives the result 1 and quantized model gives 0:

$$\forall x_i \text{ in TestSet } X : \sum_{i=1}^{m} w_i x_i < 0 \text{ and } \sum_{i=1}^{m} (w_i + \delta_i) x_i > 0$$

If floating model gives the result 0 and quantized model gives 1:

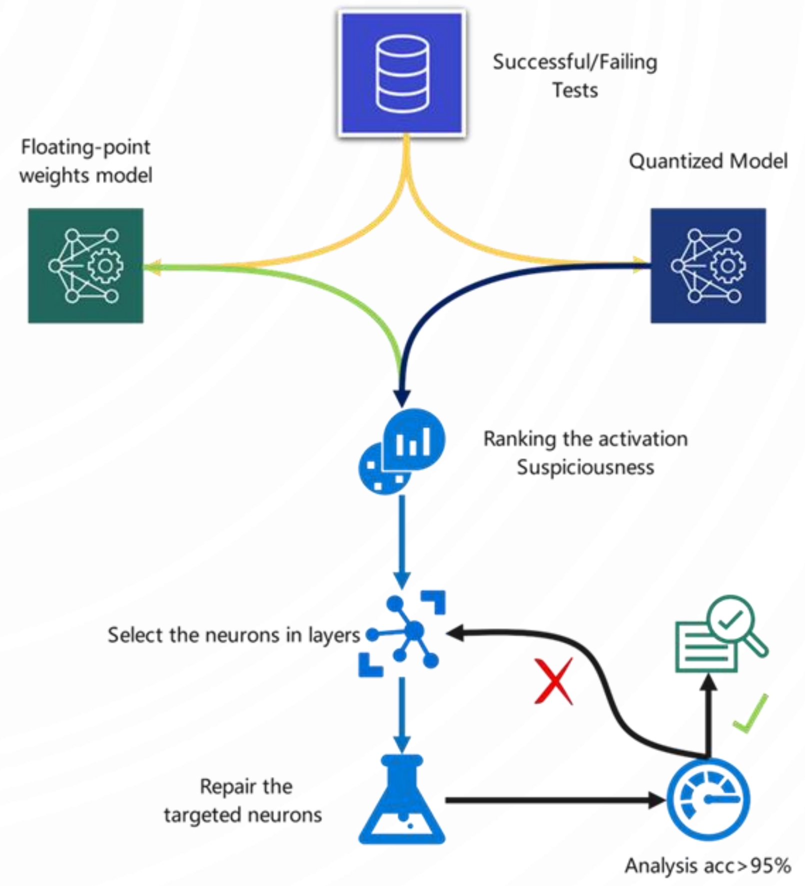
$$\forall x_i \text{ in TestSet } X : \sum_{i=1}^{m} w_i x_i > 0 \text{ and } \sum_{i=1}^{m} (w_i + \delta_i) x_i < 0$$

Overall Algorithm



Algorithm 1: Repair algorithm

```
Input: Floating-point model $F$, Quantized model $Q$, Repair set $X$, Validation
       set $V$, Number of neurons to be repaired $N$
Output: Repaired model $Q'$, Repaired model's accuracy $Acc$
Initialize $F_a[][], Q_a[][], I_n[], C^{as}[], C^{af}[], C^{ns}[], C^{nf}[]$
foreach $X$ do
    $F_a[][i] = \mathrm{getActStatus}(F, x_i)$
    $Q_a[][i] = \mathrm{getActStatus}(Q, x_i)$
    if $x[i]$ is a failing test then
        $C^{af}[i] = C^{af}[i] + |F_a[][i] - Q_a[][i]|$
    else
        $C^{as}[i] = C^{as}[i] + |F_a[][i] - Q_a[][i]|$
    end
end
$C^{nf}[] = C^{nf}[] - C^{af}[]$
$C^{ns}[] = C^{ns}[] - C^{as}[]$
$I_n[] = \mathrm{DStar}(C_n^{as}[], C_n^{as}[], C_n^{as}[], C_n^{as}[])$
$I_n[] = \mathrm{sort}(I_n[])$ // In descending order
Initialize weight of neurons $w[][]$ and the increment $\delta[][]$
foreach $neuron[i] \in I_n[]$ do
    foreach $edge[j][i] \in neuron[i]$ do
        $w[j][i] = \mathrm{getWeight}(edge[j][i])$
    end
    $\delta[][i] = \mathrm{solve}(X, w[][i])$ // Solve LP problem [3]
    foreach $edge[j][i] \in neuron[i]$ do
        $edge[j][i] = \mathrm{setWeight}(w[j][i] + \delta[j][i])$
        $Q' = \mathrm{update}(Q, edge[j][i])$
    end
    if $i >= N$ then
        break
    end
end
$Acc = \mathrm{calculateAcc}(Q', V)$
return $Q'$
```


Successful/Failing Tests
Floating-point weights model
Quantized Model
Ranking the activation Suspiciousness
Select the neurons in layers
Repair the targeted neurons
Analysis acc>95%

The University of Manchester

Experiments Baselines

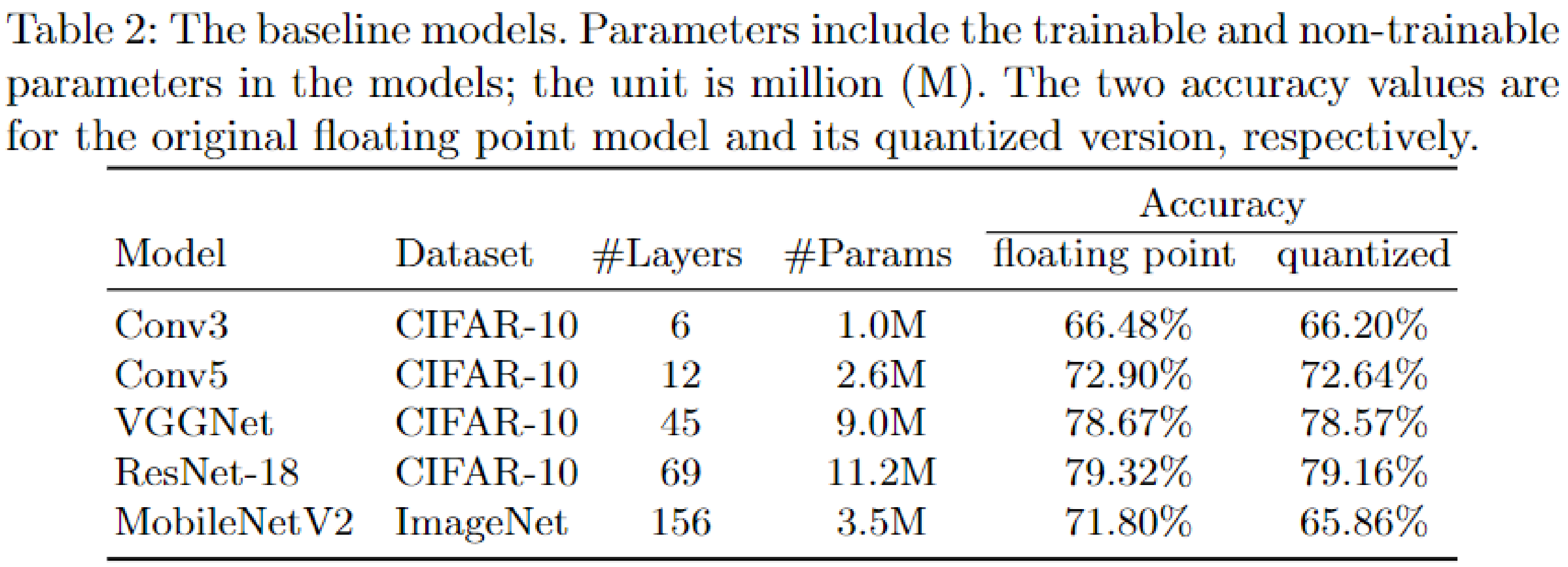


Table 2: The baseline models. Parameters include the trainable and non-trainable parameters in the models; the unit is million (M). The two accuracy values are for the original floating point model and its quantized version, respectively.

Model	Dataset	#Layers	#Params	Accuracy floating point	Accuracy quantized
Conv3	CIFAR-10	6	1.0M	66.48%	66.20%
Conv5	CIFAR-10	12	2.6M	72.90%	72.64%
VGGNet	CIFAR-10	45	9.0M	78.67%	78.57%
ResNet-18	CIFAR-10	69	11.2M	79.32%	79.16%
MobileNetV2	ImageNet	156	3.5M	71.80%	65.86%

We ran experiments on a machine with Ubuntu 18.04.6 LTS OS Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz and two Nvidia Quadro RTX 6000 GPUs.

The codes of this paper are available at: https://github.com/HymnOfLight/QNNRepair

Experiments Results

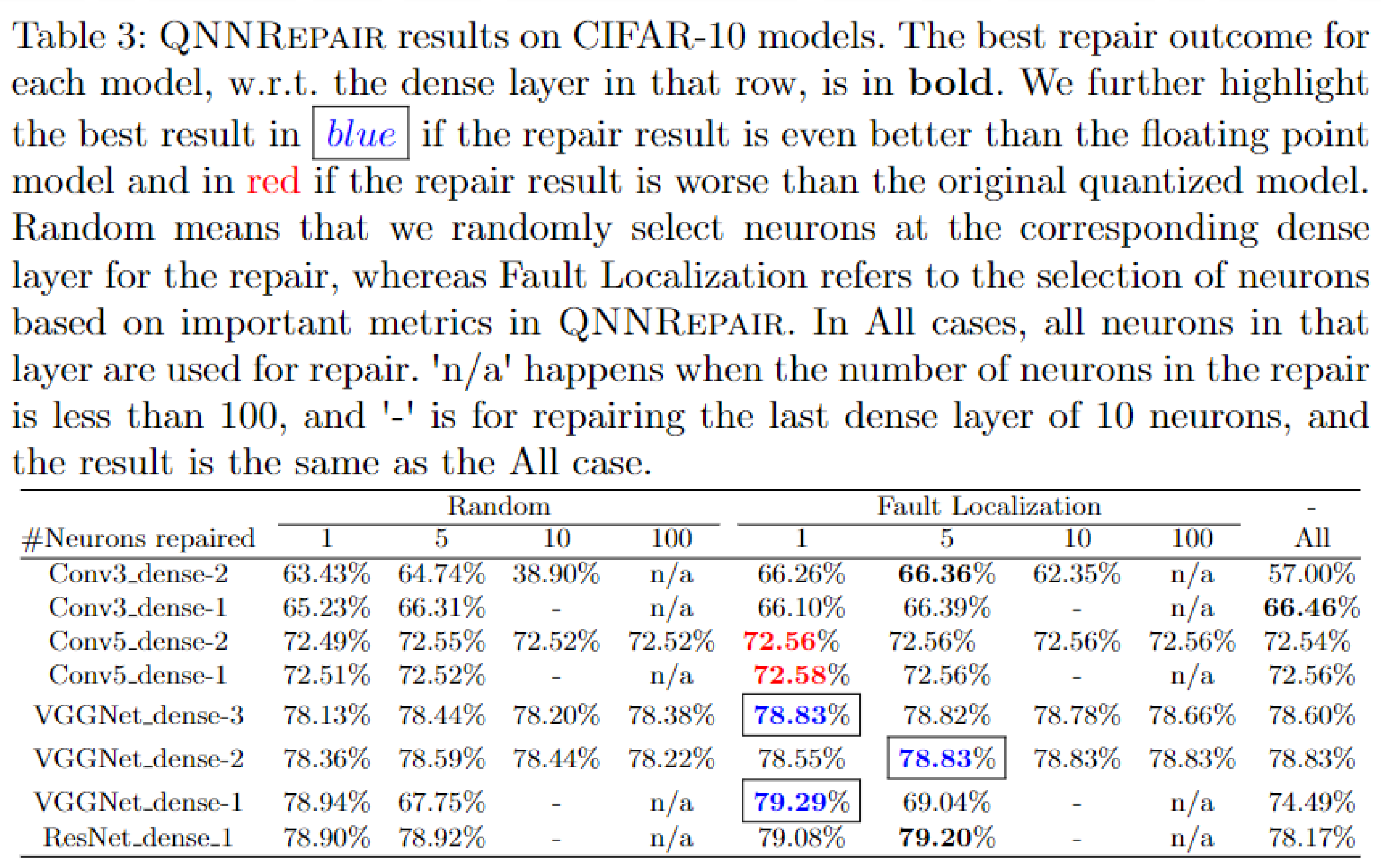


Table 3: QNNRepair results on CIFAR-10 models. The best repair outcome for each model, w.r.t. the dense layer in that row, is in **bold**. We further highlight the best result in *blue* if the repair result is even better than the floating point model and in red if the repair result is worse than the original quantized model. Random means that we randomly select neurons at the corresponding dense layer for the repair, whereas Fault Localization refers to the selection of neurons based on important metrics in QNNRepair. In All cases, all neurons in that layer are used for repair. 'n/a' happens when the number of neurons in the repair is less than 100, and '-' is for repairing the last dense layer of 10 neurons, and the result is the same as the All case.

	Random				Fault Localization				-
#Neurons repaired	1	5	10	100	1	5	10	100	All
Conv3_dense-2	63.43%	64.74%	38.90%	n/a	66.26%	**66.36%**	62.35%	n/a	57.00%
Conv3_dense-1	65.23%	66.31%	-	n/a	66.10%	66.39%	-	n/a	**66.46%**
Conv5_dense-2	72.49%	72.55%	72.52%	72.52%	**72.56%**	72.56%	72.56%	72.56%	72.54%
Conv5_dense-1	72.51%	72.52%	-	n/a	**72.58%**	72.56%	-	n/a	72.56%
VGGNet_dense-3	78.13%	78.44%	78.20%	78.38%	$\boxed{\textbf{78.83\%}}$	78.82%	78.78%	78.66%	78.60%
VGGNet_dense-2	78.36%	78.59%	78.44%	78.22%	78.55%	$\boxed{\textbf{78.83\%}}$	78.83%	78.83%	78.83%
VGGNet_dense-1	78.94%	67.75%	-	n/a	$\boxed{\textbf{79.29\%}}$	69.04%	-	n/a	74.49%
ResNet_dense_1	78.90%	78.92%	-	n/a	79.08%	**79.20%**	-	n/a	78.17%

Experiments Results

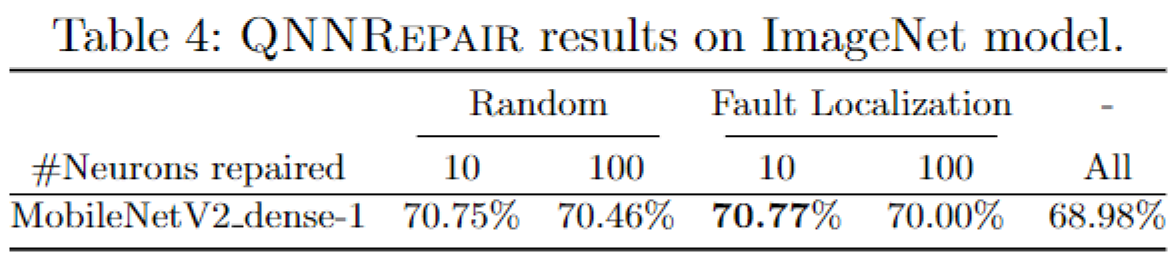


Table 4: QNNRepair results on ImageNet model.

	Random		Fault Localization		-
#Neurons repaired	10	100	10	100	All
MobileNetV2_dense-1	70.75%	70.46%	**70.77%**	70.00%	68.98%

ImageNet: We also conducted repair on the last layer for MobileNetV2 trained on the ImageNet dataset of high-resolution images. Using Euclid as the importance metric and picking 10 neurons as the correct targets achieve the best results, at 70.77%, improving the accuracy the quantized model.

The University of Manchester

Experiments Results

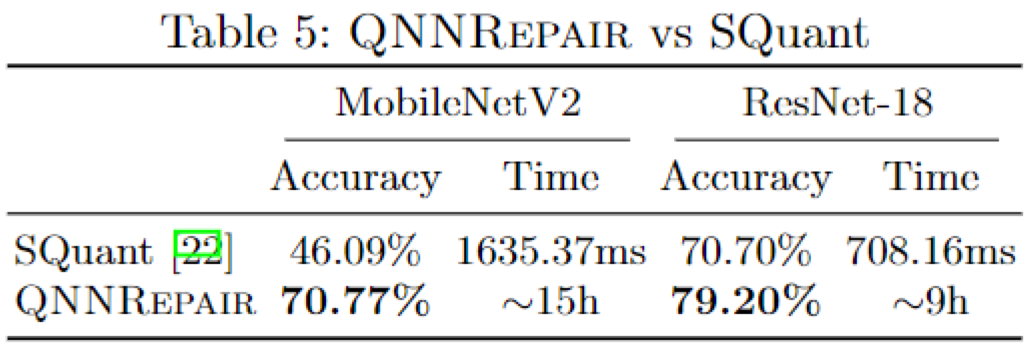


Table 5: QNNRepair vs SQuant

	MobileNetV2		ResNet-18	
	Accuracy	Time	Accuracy	Time
SQuant [22]	46.09%	1635.37ms	70.70%	708.16ms
QNNRepair	**70.77%**	~15h	**79.20%**	~9h

SQuant, a fast and accurate data-free quantization framework for convolutional neural networks.
We tested SQuant two quantized models, the same as our approach: MobileNetV2 trained on ImageNet and ResNet-18 on CIFAR-10

Guo, Cong, et al. "SQuant: On-the-Fly Data-Free Quantization via Diagonal Hessian Approximation." International Conference on Learning Representations. 2021.

MANCHESTER 1824
The University of Manchester

Repair Efficiency

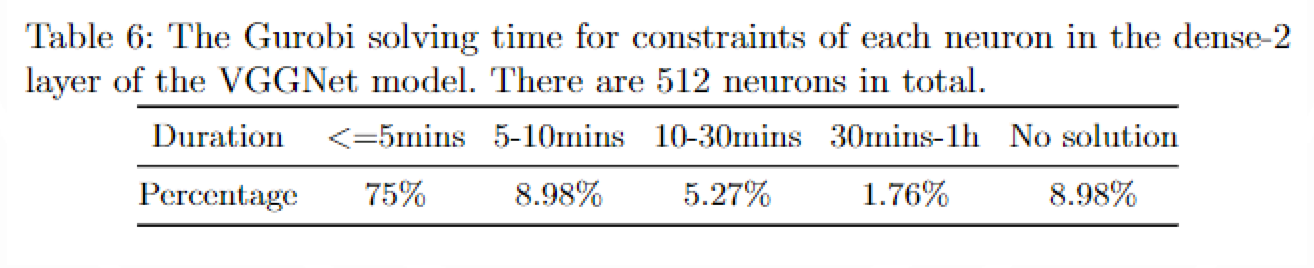


Table 6: The Gurobi solving time for constraints of each neuron in the dense-2 layer of the VGGNet model. There are 512 neurons in total.

Duration	<=5mins	5-10mins	10-30mins	30mins-1h	No solution
Percentage	75%	8.98%	5.27%	1.76%	8.98%

Table 6 measures the runtime cost when using the Gurobi to solve the values of the new weights for a neuron for our experiments on the VGGNet model. It is shown in Table 6 that 75% of the solutions were completed within 5 minutes, and less than 9% of the neurons could not be solved, resulting in a total solution time of 9 hours for a layer of 512 neurons.

MANCHESTER
1824
The University of Manchester

Conclusion and Future works

- We presented QNNRepair, a novel method for repairing quantized neural networks.
- We evaluated the importance of the neural network models and used Gurobi to get the correction for these neurons. We also compared our method with state-of-the-art techniques
- In the future we will move forward to larger datasets; we will test our tool and make it scalable for larger models and not limited to classification tasks like GPT and stable diffusion. We will find a balance between improving accuracy and computing time.
- We intend to optimize the encoding of the neural network repair problem to increase the speed of the repair solution and to solve some of the repair problems that were not previously solved

MANCHESTER
1824
The University of Manchester

Thank you!