# MicroBotNet: Low Power Neural Networks for Microrobots

*Brian Liao*

Acknowledgement

To my friends Eric, Daniel, Tushar, Angela, Vera, Jhinuk, Alan, Suchir, and Dennis who helped and supported me throughout my research and college journey. I will cheerish these memories and experiences for years to come.

To Eric for the invaluable help in this research project. I could not have done it without your help.

# MicroBotNet: Low Power Neural Networks for Microrobots

by Brian Liao

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

**Committee:**

[illegible]

Professor Kristofer Pister
Research Advisor

5/28/2020

(Date)

* * * * * * *

[illegible]

Professor Yakun Sophia Shao
Second Reader

4/30/2020

(Date)

# MicroBotNet: Low Power Neural Networks for Microrobots

Brian Liao[1]

***Abstract*— We present MicroBotNet, a neural network architecture for image classification with fewer than 1 million multiply-and-accumulate (MAC) operations. We wish to explore architectures suitable for microrobots with a goal of less than 1 µJ per forward-pass. We estimate this to be feasible with 1 million MAC operations. MicroBotNet achieves 80.47% accuracy with 740,000 MAC operations on CIFAR-10. Additionally, 60% of weights are quantized to $\{-1, 0, +1\}$ using Trained Ternary Quantization. We also evaluate MicroBotNet on our Micro Robot Dataset, which is composed of 10 image classes a microrobot may encounter such as acorns, mushrooms, and ladybugs. After applying transfer learning, MicroBotNet achieves 67.80% accuracy on the Micro Robot Dataset. Finally, we test MicroBotNet on acorn images simulating a microrobot. MicroBotNet correctly identify the acorn in 7 out of 8 cases when approaching an acorn and 15 out of 16 cases from different angles using a best of last three frames filter.**

## I. Introduction

Microrobots are miniature robots that operate at the micro-millimeter scale. Engineering microrobots is important as it help us explore the limits of scaling electronics and robotics down to the micro scale using principles from Microelectromechanical systems (MEMs). An important part of creating autonomous microrobots is to enable vision systems that help the robot with planning, object avoidance, and path-finding. Deep Learning using artificial neural networks has been effective for vision tasks. However, these neural networks are computationally expensive and consume high amounts of power, making them difficult to use in microrobot settings.

Work has been done with compression and quantization of neural networks to make them more efficient. We wish to explore the limits of these methods to see how feasible they are for microrobots, which have extremely limited battery power for computation. Our goal is to build a neural network with less than 1 µJ per forward-pass. This pushes the limits of low power neural networks and is a magnitude smaller than current commercial and research neural networks. By doing so, this has applications in other extremely low power devices such as future low power Edge and Internet of Things devices.

In this paper, we introduce **MicroBotNet**, a neural network architecture for image classification with fewer than 1 million multiply-and-accumulate (MAC) operations. We use this as an estimate benchmark for building neural networks with less than 1 µJ per forward-pass. MicroBotNet is built using three techniques for compressed and efficient neural networks. First, we use the compressed neural network model architecture from MobileNetV3 [1] as the basis for our model. Second, we apply Fast-Downsampling [2] to our model, compressing it further and significantly reducing the number of MAC operations. Finally, we apply Trained Ternary Quantization, [3] which prunes our model and makes a subset of MAC operations more efficient.

btl787@berkeley.edu

[1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

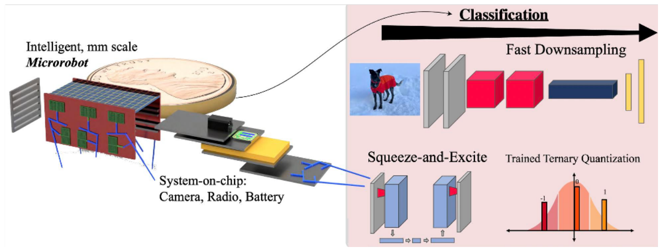


Fig. 1: Applications of MicroBotNet to microrobots. MicroBotNet does low power image classification with an efficient neural network architecture, fast downsampling, and trained ternary quantization.

We evaluate our model on CIFAR-10 and our own Micro Robot Dataset and show that our model is able to maintain accuracy and is practical for use. We also test our model on acorns images we take to ensure the model is transferable and can work in microrobot situations.

## II. Background and Related Work

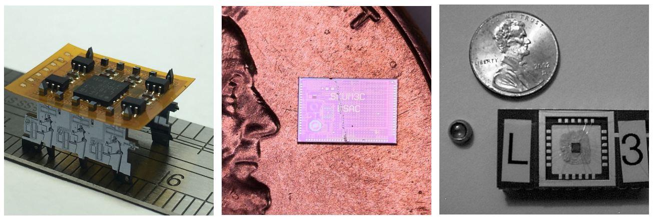
### A. Microrobots

Current examples of microrobots are heavily inspired by biological designs of insects such as walkers [4], [5] and winged flapping fliers [6]–[8]. To build on top of these developments in microrobot locomotion, vision systems are a natural evolution to assist with path finding and object avoidance.

Additional work to enable micro-scale robots and computation includes system-on-chip (SoC) breakthroughs at $\mathrm{mm}$ scale computation, sensing, and power. Chips on the scale of $\mathrm{mm}^2$ have been created for $100\,\mathrm{V}$, multi-channel power [9] and radio communication [10]. Finally, a $2\,\mathrm{mm}^3$ camera has been shown to take $128 \times 128$ greyscale images [11] and can be extended for microrobot purposes. Examples of these microrobots and components are shown in Figure 2.

### B. Neural Network Efficiency

When evaluating neural networks for efficiency, two common metrics used are the number of parameters in a neural network and the number of multiply-and-accumulate (MAC) operations in a forward pass. MACs are a more accepted



(a) Silicon Walker [5] (b) SCµM [10] (c) $\mathrm{mm}^3$ imager [11]

Fig. 2: Pieces of the micro-robot platform. Left) a silicon walking microrobot, center) a single-chip mote with integrated radio TRX, microprocessor, and sensor interface, and right) an initial $\mathrm{mm}^3$ camera capable of $128 \times 128$ pixel images.

computational-cost metric as they map to the multiply-and-accumulate computation in the partial sums for layer output maps.

SqueezeNet [12] introduced Fire Modules as a compression method in an effort to reduce the number of parameters while maintaining accuracy. Fire Modules reduce some of the $3 \times 3$ convolutions to $1 \times 1$ convolutions, resulting in 9x less parameters. MobileNetV1 [13] replaces standard convolutions with depth-wise separable convolutions. The depth-wise convolution performs spatial filtering and pointwise convolutions to generate features. Fd-MobileNet applies Fast Downsampling [2] on MobileNet for extremely computationally constrained tasks. It performs $32\times$ downsampling in the first 12 layers, dropping the computation by $4\times$ at a 5% accuracy loss. MobileNetV3 [1] uses neural architecture search, optimizing for efficiency to design the model. Other improvements include 'hard' activation functions (h-swish and h-sigmoid) [14], inverted residuals and linear bottlenecks [15], and squeeze-and-excite layers [16] that extract spatial and channel-wise information. SqueezeNext [17] improves on depth-wise separable convolutions with low rank separable filters for higher arithmetic intensity (ratio of compute to memory operations). They note that arithmetic intensity serves as a better metric than MACs for performance of energy efficiency since it additionally measures hardware utilization. This can guide future design for lower power neural network models.

Trained Ternary Quantization [3] reduces weight precision to 2-bit ternary values with scaling factors, and achieve no accuracy loss. Hessian Aware Quantization [18] recognizes that using low precision leads to accuracy degradation. Instead, they propose using mixed-precision, where layers can be selected for quantization based on their second order Hessian eigenvalues. These eigenvalues provides information on the sensitivity of a layer to quantization. Eyeriss [19] provides a taxonomy of execution dataflows in neural networks and shows that row stationary dataflow is more efficient than output stationary, weight stationary, and no local reuse dataflows.

Benchmarking from a $45\mathrm{nm}$ process [20], shrinking process nodes and decreased bit precision enable a MAC cost that approaches $1\mathrm{pJ}$. Targeting $1\mu\mathrm{J}$ per forward-pass, we combine these advancements into a new network with $<1$ million MACs.

| Input | Operator | exp size | # out | SE | s |
|---|---|---|---|---|---|
| $32^2 \times 3$ | conv2d, $3 \times 3$ | - | 16 | - | 2 |
| $16^2 \times 16$ | bneck, $3 \times 3$ | 72 | 24 | No | 2 |
| $8^2 \times 24$ | bneck, $5 \times 5$ | 96 | 40 | Yes | 2 |
| $4^2 \times 40$ | bneck, $5 \times 5$ | 240 | 40 | Yes | 1 |
| $4^2 \times 40$ | bneck, $5 \times 5$ | 120 | 48 | Yes | 1 |
| $4^2 \times 48$ | bneck, $5 \times 5$ | 144 | 48 | Yes | 1 |
| $4^2 \times 96$ | bneck, $5 \times 5$ | 288 | 96 | Yes | 2 |
| $2^2 \times 96$ | bneck, $5 \times 5$ | 576 | 96 | Yes | 1 |
| $2^2 \times 96$ | bneck, $5 \times 5$ | 576 | 96 | Yes | 1 |
| $2^2 \times 96$ | bneck, $5 \times 5$ | 576 | 96 | Yes | 1 |
| $2^2 \times 96$ | bneck, $5 \times 5$ | 576 | 96 | Yes | 1 |
| $2^2 \times 96$ | conv2d, $1 \times 1$ | - | 576 | Yes | 1 |
| $2^2 \times 576$ | pool, $2 \times 2$ | - | - | - | 1 |
| $1^2 \times 576$ | conv2d, $1 \times 1$ | - | 1024 | - | 1 |
| $1^2 \times 1024$ | linear | - | 10 | - | 1 |

TABLE I: Model specification of MicroBotNet (SE indicates if a Squeeze-And-Excite is used, s indicates the stride used).

## III. MicroBotNet

### A. Architecture

MicroBotNet adapts the neural network architecture of MobileNet V3 [1] to achieve low power image classification. MicroBotNet uses efficient mobile bottleneck layers, which are built with linear bottlenecks and inverted residuals, and squeeze and excite attention modules that help extract spatial and channel-wise information. MicroBotNet also uses hard activation functions that are more hardware efficient.

The overall network architecture is shown in Table I.

*1)* ***Linear Bottlenecks and Inverted Residuals****:* Bottleneck layers are based on the idea that using low dimensional tensors reduces the computation and number of MAC operations required. However, filtering cannot extract a lot of information from low dimensional tensors as a lot of information is lost. Linear bottlenecks are designed to have the input activation begin in a low dimensional tensor, expand to high dimensional tensor which we can apply a depthwise convolution to, and shrink back down to a low dimension tensor. The inverted residual applies skip connections between different bottleneck layers to help apply gradient flow in the network. The residual is "inverted" because we have residual connections among layers of a low dimensional tensor, a later high dimensional tensor, and another low dimensional tensor. Normal residual networks have a residual connection between layers of a high dimensional tensor, a later low dimensional tensor, and another high dimensional tensor.

This bottleneck module is shown in Table II.

*2)* ***Squeeze-and-Excite****:* The goal of squeeze and excite modules is to globally embed channel-wise dependencies in layers. Convolutions extract spatial wise dependencies using filters that stride along a local field. To embed channel wise dependencies, we encode the channels into a vector statistic $z_c$ with a **Squeeze** operation which can be done with global average pooling.

| Operator | Input | Output | Purpose |
|---|---|---|---|
| conv2d, $1 \times 1$ | C | E | Uncompress Data |
| bnorm | - | - | - |
| hswish | - | - | - |
| conv2d, $F \times F$ | E | E | Filter Data |
| hswish | - | - | - |
| squeeze, | E | E | Embedded Channel Data |
| hswish | - | - | - |
| conv2d, $1 \times 1$ | E | K | Compress Data |
| bnorm | - | - | - |

TABLE II: MobileNet V3 Bottleneck Layers. Input and Output are the number of channels where C, E, K are channel sizes and E >> C, K. F is a filter size. "squeeze" is a Squeeze-and-Excite module.

$$z_c = \boldsymbol{F}_{sq}(\boldsymbol{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j) = \text{avgpool}(u_c) \tag{1}$$

The second step is an **Excite** operation that applies self-attention from the statistic. This is done with a sigmoid that gates the statistic and is parameterized by $\boldsymbol{W1}$ , $\boldsymbol{W2}$, and a non-linear activation. Doing so provides a weighting to the input tensor that shows what information to show attention to.

$$s = \boldsymbol{F}_{ex}(\boldsymbol{z}, \boldsymbol{W}) = \sigma(\boldsymbol{W2} \cdot \text{h-swish}(\boldsymbol{W1} \cdot \boldsymbol{z})) \tag{2}$$

The squeeze-and-excite module is able to embed global channel information in the layer and is shown in Table III.

*3) **Hard Activation Functions**:* Hard activation functions are functions that are efficient to calculate in floating point hardware. Nonlinearities, such as sigmoid defined below,

$$\sigma(x) = \frac{e^x}{1 + e^x} \tag{3}$$

require complex numerical precision to calculate. ReLU,

$$\text{ReLU}(x) = \max(x, 0) \tag{4}$$

by comparison, requires only a hardware comparison to calculate. ReLU6,

$$\text{ReLU6}(x) = \min(\max(x, 0), 6) \tag{5}$$

caps the output activation at 6, making the output activation representable when quantized to 3 bits. MobileNet V3 introduced the use of h-sigmoid and h-swish where the h stands for "hard" activation function. h-sigmoid is defined as

$$\text{h-sigmoid}(x) = \frac{\text{ReLU6}(x+3)}{6} \tag{6}$$

It approximates sigmoid without $e^x$, allowing for self-attention gating in the squeeze-and-excite module.

Swish is an activation function similar to ReLU in that its output is approximately 0 when the input is less than 0, and the input value when the input is greater than 0. Swish,

| Operator | Input | Output | Purpose |
|---|---|---|---|
| pool, $1 \times 1$ | $H \times W \times E$ | $1^2 \times E$ | Global Embedding |
| linear | $1^2 \times E$ | $1^2 \times C$ | Embedded Channels |
| relu | - | - | - |
| linear | $1^2 \times C$ | $1^2 \times E$ | Embedded Channels |
| hsigmoid | - | - | Self-Attention |

TABLE III: Squeeze-and-Excite modules. Input and Output are tensors where H, W are the current input dimensions, E is the expanded channel size, and C is the input channel size of the bottleneck layer.

however, is smooth and differentiable at 0, assisting gradient flow. It is defined as

$$\text{swish}(x) = x \cdot \sigma(x) \tag{7}$$

and h-swish is defined as

$$\text{h-swish}(x) = x \cdot \text{h-sigmoid}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} \tag{8}$$

MicroBotNet uses h-swish for nonlinearity activations in our network and h-sigmoid for self-attention gating.

## B. Fast Downsampling

To reduce the number of MAC operations, we use fast-downsampling as introduced in Fd-MobileNet [2]. Fast Downsampling downsamples our network early by doing striding in the first layers of our network. This reduces the width and height feature dimension inputs by a half and decreases the number of MAC operations by a fourth. The number of channels is usually doubled or increased to compensate but gives a savings of half the number of MAC operations. When applied early in our network, this drastically reduces the number of MAC operations required.

MicroBotNet does $16\times$ fast-downsampling in the first 8 layers. To compensate, we add 4 bottleneck $5 \times 5$ layers with an expand size of 576 at the end of our network for suitable network information capacity. We also include the width multiplier $\alpha$ for the number of channels, which allows the model to generalize based on one's MAC computation needs. Width multipliers of $\times 0.25$, $\times 0.50$, and $\times 1.00$ are included as reference.

## C. Quantization

In addition, we quantize our model to take advantage of reduced energy costs when retrieving memory access from a fixed precision of numbers. This also allows the network to have an efficient access pattern when using a weight stationary dataflow. We use trained ternary quantization [3] which quantizes weights in each layer to a negative value, zero, or a positive value $\{-W_l^n, 0, +W_l^p\}$. Values that are quantized to 0 are also pruned and do require a Multiply-and-Accumulate operation, reducing our MAC count. To quantize, we use a threshold value $\Delta_l$ defined as

$$\Delta_l = t \times \max(|\tilde{w}_l|) \tag{9}$$

where $\tilde{w}_l$ is our full precision weights for each layer. This gives the final quantized weight

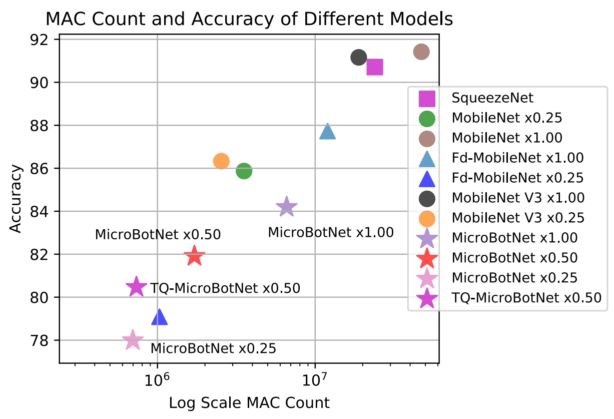


Fig. 3: Tradeoff of MAC Count and Accuracy of different models on CIFAR-10. TQ-MicroBotNet ×0.50 improves on FD-MobileNet ×0.25 while continuing the trend of accuracy towards low-MACs.

$$w_l^t = \begin{cases} +W_l^p & \tilde{w}_l > \Delta_l \\ 0 & |\tilde{w}_l| \leq \Delta_l \\ -W_l{}^n & \tilde{w}_l < -\Delta_l \end{cases} \tag{10}$$

After applying trained ternary quantization to our model, we noticed $-W_l{}^n \approx -1$ and $+W_l^p \approx +1$. Multiplying an input activation by a weight of 1 is the input activation itself, and multiplying an input activation by a weight of $-1$ is a bit flip of the sign bit in floating point. Both of which are hardware efficient operations. To take advantage of these, we used a mask to requantize $\{-W_l{}^n,\ 0,\ +W_l^p\}$ to ternary quantization $\{-1,\ 0,\ +1\}$.

## IV. Experimental Setup and Results

### A. CIFAR-10

*1) Experimental Setup:* We evaluate the accuracy of different energy-efficient neural network models on the CIFAR-10 dataset [21] using a 12GB Nvidia K80 GPU. The number of parameters and MAC operations are calculated with THOP [22]. Each model is trained with full 32-bit precision. We use standard pre-processing including random cropping, random horizontal flipping, and normalization of training and testing images. We process images with a batch size of 256. For each model, we use stochastic gradient descent with 0.9 momentum, 5e-4 weight decay, a learning schedule with a learning rate of 0.1, and a decay of 0.1 every 50 epochs for 200 epochs. We benchmark against SqueezeNet, MobileNet, FD-MobileNet, and MobileNet V3 with width multipliers of ×0.25 and ×1.00. We validate against our model, **MicroBotNet**, with width multipliers of ×0.25, ×0.50, and ×1.00.

We also evaluated applying ternary quantization to our MicroBotNet ×0.50 model. Our model uses several 1 × 1 convolutions that, when quantized, do not have enough information bandwidth to maintain accuracy. Instead, we quantize all fully connected and convolution layers with number of parameters above 5000, not including the initial convolution and final fully connected layers.

| Model | Top-1 | MAC | Parameters |
|---|---|---|---|
| MobileNet V3 ×0.25 | 86.33 | 2,540,820 | **124,050** |
| MobileNet ×0.25 | 85.87 | 3,539,456 | 215,642 |
| **MicroBotNet ×0.50** | 81.91 | 1,715,024 | 538,138 |
| **TQ-MicroBotNet ×0.50** | 80.47 | 735,402 | 230,753 |
| Fd-MobileNet ×0.25 | 79.09 | 1,029,888 | 128,730 |
| **MicroBotNet ×0.25** | 77.99 | **697,662** | 160,162 |

TABLE IV: Top-1 is accuracy on CIFAR-10. Comparison of MicroBotNet ×0.25 and ×0.50 and Ternary Quantized MicroBotNet ×0.50 with similar architectures.

| Model | Top-1 | MAC | Parameters |
|---|---|---|---|
| MobileNet ×1.00 | 91.42 | 47,187,968 | 3,217,226 |
| MobileNet V3 ×1.00 | 91.16 | 18,891,842 | 1,518,594 |
| SqueezeNet | 90.71 | 23,902,388 | **730,314** |
| Fd-MobileNet ×1.00 | 87.73 | 11,983,872 | 1,886,538 |
| **MicroBotNet ×1.00** | 84.19 | **6,597,218** | 2,044,298 |

TABLE V: Top-1 is accuracy on CIFAR-10. Comparison of MicroBotNet ×1.00 with similar architectures.

*2) Results:* Ternary Quantized MicroBotNet (TQ-MicroBotNet) ×0.50 achieves 80.47% accuracy on CIFAR-10 while only using 735,402 MACs. This outperforms the previous work by 1.38% in FD-MobileNet ×0.25, which achieves 79.09% accuracy with 1,029,888 MACs. Here, it is clear that our model is able to maintain accuracy and achieve our goal of staying under 1 million MACs. Our work represents a further step in the trend of low-power classification as shown in Figure 3.

TQ-MicroBotNet ×0.50 is able to quantize parameters by 60% where 57% of the model is pruned to zero. This makes the network efficient for processor elements that do not have to evaluate pruned MAC operations. The distribution of quantized weights can be seen in Figure 5.

### B. Micro Robot Dataset

*1) Experimental Setup:* To further evaluate our results, we created a dataset selecting for classes of objects a microrobot may encounter in the wild. These images were pulled from Tiny-ImageNet [23], which is 200 classes of 500 training images, 50 validation images, and 50 test images, at 64×64 resolution. The Micro Robot Dataset is 10 classes of **acorns, bees, black widows, frogs, ladybugs, mushrooms, nails, sandals, socks, and tarantulas** with 500 training images and 50 test images at 32×32 resolution. Reference images can be seen in Figure 4.

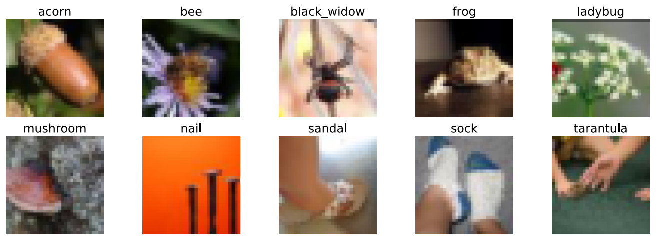


Fig. 4: The Micro Robot Dataset is composed of acorns, bees, black widows, frogs, ladybugs, mushrooms, nails, sandals, socks, and tarantulas, 10 classes a microrobot may encounter.

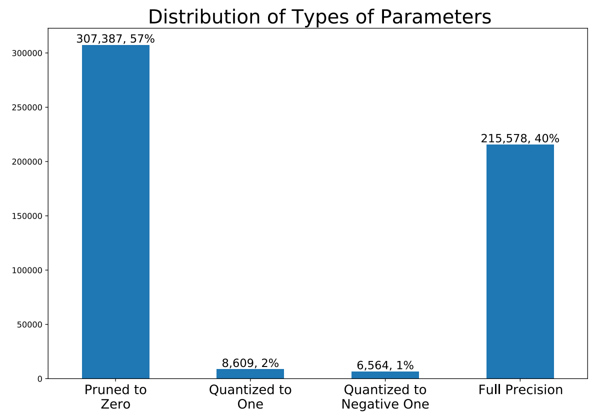


Fig. 5: Distribution of the types of parameters in Ternary Quantized MicroBotNet. 57% of parameters are pruned to zero and 60% are quantized to efficient hardware operations.

| Model | Top-1 | MAC | Parameters |
|---|---|---|---|
| **TQ-MicroBotNet $\times$0.50** | 67.80 | 735,402 | 230,753 |
| Fd-MobileNet $\times$0.25 | 65.60 | 1,029,888 | **128,730** |
| **MicroBotNet $\times$0.25** | 65.20 | **697,662** | 160,162 |

TABLE VI: Top-1 is accuracy on Micro Robot Dataset.

We use a 12GB Nvidia K80 GPU and preprocess with random cropping, random horizontal flipping, and normalization of training and testing images at a batch size of 128. We do coarse grain transfer learning on the Micro Robot Dataset, using stochastic gradient decent with 0.9 momentum, 5e-4 weight decay, and a learning schedule with a learning rate of $1e-3$ and a decay of 0.1 every 25 epochs for 100 epochs.

*2) Results:* With coarse grain transfer learning, TQ-MicroBotNet $\times$0.50 achieves an accuracy of 67.80% on the Micro Robot Dataset. For comparison, Fd-MobileNet $\times$0.25 achieves 65.60% accuracy. Future work to improve this model can be done by collecting more data such as using a larger dataset like IP102 [24] which contains 75,222 images of 102 insect classes.

### C. Emulating Vision Tasks for Microrobots

To evaluate the effectiveness of our network for use in microrobot tasks, we simulated a scenario of identifying acorns with images from a microrobot perspective. This would help us evaluate the transferability from datasets such as CIFAR-10 and our Micro Robot Dataset to real world microrobot applications. We took 10 images of acorns simulating a microrobot approach in four scenarios in the wild:

1) Single Acorn on Pavement
2) Multiple Acorns on Pavement
3) Single Acorn in Grass
4) Multiple Acorns in Grass

All images were taken using an iPhone 8 and cropped and downsampled to 32$\times$32. Our sequences of acorns were a combination of simulating a microrobot approaching an acorn and a microrobot rotating around an acorn for multiple perspectives and angles. Four images frames were used to simulate approaching an acorn and six image frames were use to simulate rotating around the acorn. Examples of image sequences for each class can be seen in Figure 7 and Figure 8.

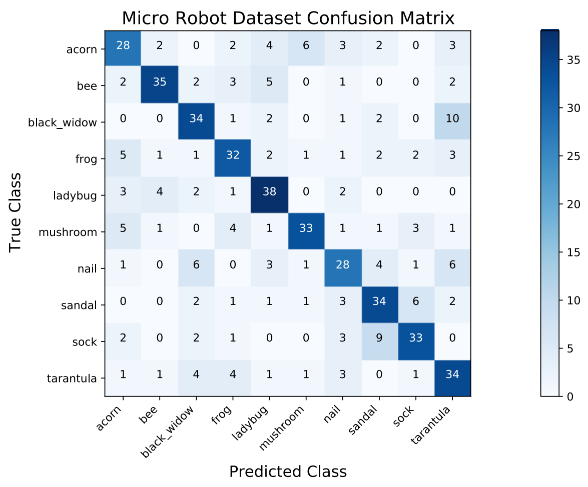


Fig. 6: Confusion Matrix of TQ-MicroBotNet $\times$0.50 on the Micro Robot Dataset. TQ-MicroBotNet $\times$0.50 accurately predicts the correct class. Predicting tarantulas for black widows is the largest misclassified class.

One additional difference from computer vision benchmarks is that a microrobot in the wild can move towards and around an object and have higher confidence in identifying an object. To take this into account, we use a simple filter that would the most common class of the last three images taken.

Using a best of last three image filter, MicroBotNet is able to accurately identify the acorn in 7 out 8 cases when approaching an acorn. MicroBotNet also correctly identifies the acorn in 15 out of 16 cases when viewing the acorn from different angles. Accuracy results for each scenario are included in Table VII.

In most cases, our network is able to identify the acorn from taking the best of the three filter. The most successful class was Multiple Acorns on Pavement where the brown color of the acorn contrasts with the flat gray pavement

| Scenario | Total Accuracy (10 Frames) | Approach Best of 3 (4 Frames) | Rotating Best of 3 (6 Frames) |
|---|---|---|---|
| Single Acorn on Pavement | 7/10 | 2/2 | 3/4 |
| Multiple Acorns on Pavement | 8/10 | 2/2 | 4/4 |
| Single Acorn in Grass | 6/10 | 1/2 | 4/4 |
| Multiple Acorns in Grass | 7/10 | 2/2 | 4/4 |
| Total | 28/40 | 7/8 | 15/16 |

TABLE VII: Accuracy of MicroBotNet identifying acorns, simulating approaching an acorn with best of three filter, and rotating around and showing different perspectives of an acorn with best of three filter.

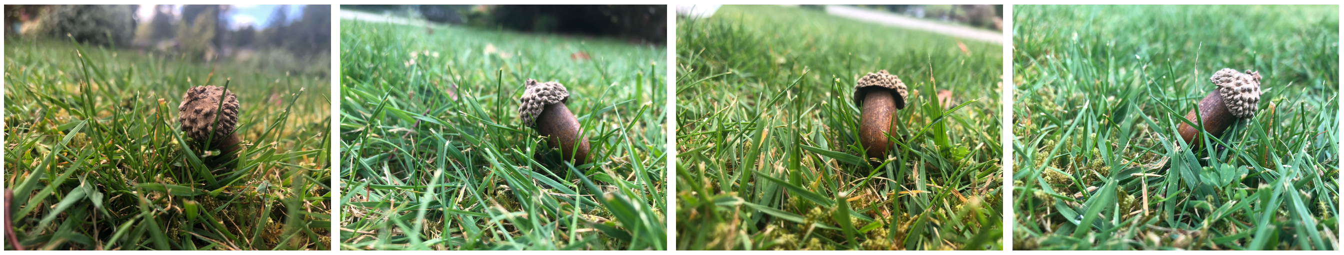


Fig. 7: Images of a single acorn in grass from different angles.

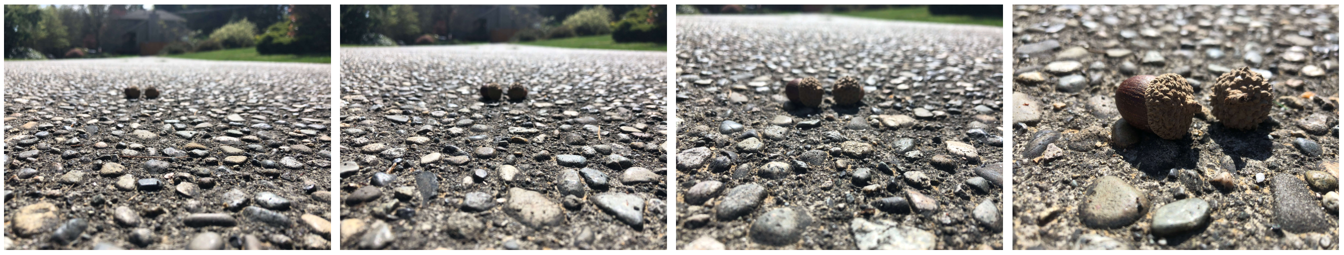


Fig. 8: Simulating an approach of a microrobot towards multiple acorns on pavement.

making it a clear target to identify. The two misclassied classes were misidentified as a nail and a bee. The first misclassified image was also taken from the furthest position away on approach, making it difficult to identify. The hardest class to identify was Single Acorn in Grass. In this scenario, the grass can cover the acorn and the grass environment makes it similar to other classes such as mushrooms and ladybugs. The four misclassified classes were two ladybugs, one mushroom, and one bee.

Another factor we considered is the false positive rate of the network to ensure it is not biased to acorns. To test this, we added 40 images from each of the other classes to create a balanced dataset with acorns and all other image classes. Our network's true positive rate was 70% with 28 out 40 images correctly classified and the false positive rate was 6.7% with 24 out of the 360 images misclassified as acorns. Our model is therefore accurate without bias towards acorns.

Overall, MicroBotNet with a best of three filter would allow a microrobot to reliably detect objects running in the wild.

## V. Conclusion

We present MicroBotNet, a neural network architecture capable of image classification with under 1 million multiply-and-accumulate (MAC) operations, approximately less than 1 µJ per forward-pass. Our network is designed for micro-robots and is an order of magnitude smaller than current research and industrial neural networks. This shows that it is possible to design performant neural network with significant efficiency constraints. MicroBotNet is evaluated on CIFAR-10 and achieves 80.47% accuracy with 740,000 MACs and 67.80% accuracy on our Micro Robot Dataset. To demonstrate that this network is practical for microrobot use, we tested MicroBotNet on images of acorns simulating a microrobot. MicroBotNet correctly identify the acorn in 7 out of 8 cases of approaching an acorn and 15 out of 16 cases for different angles with a best of last three frames filter. This demonstrates our contribution of practical and highly accurate neural networks that can be used in microrobots and other lower power IoT devices.

## References

[1] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1314–1324.

[2] Z. Qin, Z. Zhang, X. Chen, C. Wang, and Y. Peng, "Fd-mobilenet: improved mobilenet with a fast downsampling strategy," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 1363–1367.

[3] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[4] X. Zhang, J. Zhao, Q. Zhu, N. Chen, M. Zhang, and Q. Pan, "Bioinspired aquatic microrobot capable of walking on water surface like a water strider," *ACS applied materials & interfaces*, vol. 3, no. 7, pp. 2630–2636, 2011.

[5] D. S. Contreras, D. S. Drew, and K. S. Pister, "First steps of a millimeter-scale walking silicon robot," in *2017 19th International Conference on Solid-State Sensors, Actuators and Microsystems (TRANSDUCERS)*. IEEE, 2017, pp. 910–913.

[6] R. J. Wood, "The first takeoff of a biologically inspired at-scale robotic insect," *IEEE transactions on robotics*, vol. 24, no. 2, pp. 341–347, 2008.

[7] Y. Zou, W. Zhang, and Z. Zhang, "Liftoff of an electromagnetically driven insect-inspired flapping-wing robot," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1285–1289, 2016.

[8] N. T. Jafferis, E. F. Helbling, M. Karpelson, and R. J. Wood, "Untethered flight of an insect-sized flapping-wing microscale aerial vehicle," *Nature*, vol. 570, no. 7762, p. 491, 2019.

[9] J. S. Rentmeister, K. Pister, and J. T. Stauth, "A 120-330 V, sub-$\mu$A, optically powered microrobotic drive IC for DARPA SHRIMP," in *GOMACTech*, 2020.

[10] F. Maksimovic, B. Wheeler, D. C. Burnett, O. Khan, S. Mesri, I. Suciu, L. Lee, A. Moreno, A. Sundararajan, B. Zhou, R. Zoll, A. Ng, T. Chang, X. Villajosana, T. Watteyne, A. Niknejad, and K. S. J. Pister, "A crystal-free single-chip micro mote with integrated 802.15.4 compatible transceiver, sub-mw ble compatible beacon transmitter, and cortex m0," in *2019 Symposium on VLSI Circuits*, 6 2019, pp. C88–C89.

[11] J. Choy, "A $10\mu$J/frame $1\text{mm}^2$ $128 \times 128$ cmos active image sensor," Master's thesis, University of California, Berkeley, 2003.

[12] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[14] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[15] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[16] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[17] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "Squeezenext: Hardware-aware neural network design," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1638–1647.

[18] Z. Dong, Z. Yao, Y. Cai, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq-v2: Hessian aware trace-weighted quantization of neural networks," *arXiv preprint arXiv:1911.03852*, 2019.

[19] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[20] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[21] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, vol. 55, 2014.

[22] "Thop: Pytorch-opcounter." [Online]. Available: https://github.com/Lyken17/pytorch-OpCounter/

[23] L. Yao and J. Miller, "Tiny imagenet classification with convolutional neural networks," *CS 231N*, vol. 2, no. 5, p. 8, 2015.

[24] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8787–8796.