# Deployment of SE-SqueezeNext on NXP BlueBox 2.0 and NXP i.MX RT1060 MCU

Ravi Teja N.V.S Chappa
*Electrical and Computer Engineering*
*Purdue School of Engineering and Technology*
Indianapolis, USA
nagchapp@iupui.edu

Mohamed El-Sharkawy
*Electrical and Computer Engineering*
*Purdue School of Engineering and Technology*
Indianapolis, USA
melshark@iupui.edu

***Abstract*—Convolution neural system is being utilized in field of self-governing driving vehicles or driver assistance systems (ADAS), and has made extraordinary progress. Before the CNN, conventional AI calculations helped ADAS. Right now, there is an incredible investigation being done in DNNs like MobileNet, SqueezeNext & SqueezeNet. It improved the CNN designs and made it increasingly appropriate to actualize on real-time embedded systems. Due to the model size complexity of many models, they cannot be deployed straight away on real-time systems. The most important requirement will be to have less model size without a tradeoff with accuracy. Squeeze-and-Excitation SqueezeNext which is an efficient DNN with best model accuracy of 92.60% and with least model size of 0.595MB is chosen to be deployed on NXP BlueBox 2.0 and NXP i.MX RT1060. This deployment is very successful because of its less size and better accuracy. The model is trained and validated on CIFAR-10 dataset.**

***Index Terms*—Squeeze-and-Excitation SqueezeNext architecture(SE-SqueezeNext), Convolution Neural Networks (CNN), Deep Neural Networks (DNN), SqueezeNext, SqueezeNet, CIFAR-10, NXP BlueBox 2.0, NXP i.MX RT1060 MCU.***

## I. Introduction

The majority of the applications continuously, for example, computer vision, mechanical autonomy, image recognition and grouping of images [10], autonomous vehicles and ADAS have been changed with assistance of Deep Neural Networks. This has been made conceivable by experiencing profound research right now the previous decade with the accessibility of all the more preparing information, and for preparing and approval having quicker equipment. Be that as it may, not incredible measure of work is done in parts of model size and speed. There is a drawback to DNNs that it require more spending plan of assets that alludes to more calculation and memory assets. Most as of late, DNN achieved a befuddling benchmark of exactness at 99% with GPipe [17].

In this paper, we have used Squeeze-and-Excitation SqueezeNext which is abbreviated as SE-SqueezeNext. This is modified version of SqueezeNext. The modifications includes the addition of SE block(Squeeze and Excitation block), Relu inplace activation, learning rate scheduling along with nestrov, decay and momentum implemented with SGD optimizer. These changes are respect to CIFAR-10 dataset [16] so that the model size is reduced without affecting the accuracy. The architectural representation of SE-SqueezeNext is shown in Figure 1.

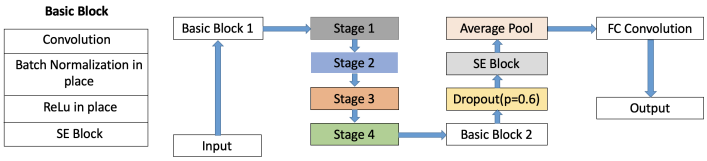



Fig. 1: Illustration of Basic Block (left) and Squeeze-and-Excitation SqueezeNext architecture.

In applications, for example, ADAS frameworks, image recognition and so forth we need a system that is straightforward and ought to be adaptable to make the application work quicker. One of those structures is RTMaps. RTMaps V4.5.0 makes the programming run on installed targets. Along these lines, one of those high computational targets is NXP Bluebox 2.0. The RTMaps installed accessible on NXP Bluebox 2.0 causes us to create applications, for example, ADAS frameworks and so on. NXP Bluebox 2.0 is an improvement stage that gives an essential presentation, utilitarian security to structure an autonomous vehicle. NXP Bluebox 2.0 comprises of S32V234 for vehicle vision and sensor combination microchip, the LS2084A implanted PC processor and the S32R27 radar microcontroller. The calculations created in have PC are conveyed in NXP Bluebox 2.0 utilizing a SSL connection.

NXP i.MX RT1060 is highest performing Arm Cortex-M7 MCU. It has industry's lowest dynamic power with an integrated DC-DC converter and operates at speeds up to 600 MHz to provide high CPU performance and best real-time response. It is supported by NXP's MCUXpresso Software's and tools. It can compute computer vision algorithms with low latency. The eIQ machine learning software helps us developing machine learning algorithms in i.MX RT1060.

The paper is organized as below. In section II, we discuss about the previous efforts which are related our work. Followed by section III describes the implementation SE-SqueezeNext on the target hardware. Section IV explains the hardware and software used for deployment of the

This is the author's manuscript of the work published in final edited form as:

Chappa, R. T. N. V. S., & El-Sharkawy, M. (2020). Deployment of SE-SqueezeNext on NXP BlueBox 2.0 and NXP i.MX RT1060 MCU. 2020 IEEE Midwest Industry Conference (MIC), 1, 1–4. https://doi.org/10.1109/MIC50194.2020.9209612

architecture. Section V shows the experimental results obtained after successful deployment using CIFAR-10 dataset. In section VI, lastly conclusions are made in the paper.

## II. Previous Work

There have been numerous structures presented after Alexnet. A portion of those systems are VGG, Inception [21], [22] and so forth. These are more precise than Alexnet yet in addition confounded. So as to execute these structures continuously is beyond the realm of imagination due to the high model size. At that point, there has been a great deal of dynamic research proceeding to create models that are good to convey continuously gadgets without settling on exactness. Out of which, two strategies are portrayed in building up these little models. One is building up a little model without any preparation or packing a huge system. Quantization, hashing [1], Pruning, vector quantization and Huffman encoding and so on. These techniques are used to compress a huge system. Systems like Mobilenet V1 and Mobilenet V2 are little models that are created without any preparation. These systems didn't concentrate on speed. Squeezenet and Squeezenext are presented which are little in size as well as concentrated on speed also. There are a lot of new systems like [19],[20],[21] they are created utilizing various convolutions like gathered convolutions and a mix of convolutions shaping remaining sort associations and so forth [22] gives a few bits of knowledge into different applications with RTMaps and BlueBox 2.0.

## III. Deployment of SE-SqueezeNext

### A. SE-SqueezeNext on NXP BlueBox 2.0

The python part in RTMaps will permit us to create and incorporate PC vision calculations for ADAS applications like Image arrangement, traffic sign recognition, and driving assistance and so on. The python segment in RTMaps has an editorial manager in it that permits clients to make, create and convey their python contents. Right now, are three principle capacities that are essential to know so as to actualize clients python content in equipment. Birth(), Core() and Death() are the three capacities that are accessible in the proofreader. Birth() is executed once toward the starting to introduce and set up the code. Core() is a capacity that runs in an unbounded loop. Along these lines, the client's code can be characterized right now permits code to run persistently. Demise() is characterized at the end and it is considered when the program is ended.

The python component in RTMaps is appeared in the figure 2. This structure of composing code makes it simpler for the client to prototyping and building up their own code as for the application. When the scripting is done, the client can utilize the RTMaps Embedded to run their application on the Bluebox platform. Figure-5 shows the flowchart of RTMaps arrangement with Bluebox 2.0. The association between the have pc and the objective Bluebox is TCP/IP. In the wake of interfacing with have pc, the client can check right COM ports in the gadget administrator. At that point client should arrangement Teraterm for LS2 interface also, S32V interface. Right now, classifier is prepared as it were in GPU yet tried in NXP Bluebox 2.0.

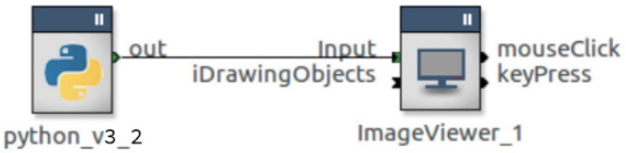



Fig. 2: The python component in RTMaps.

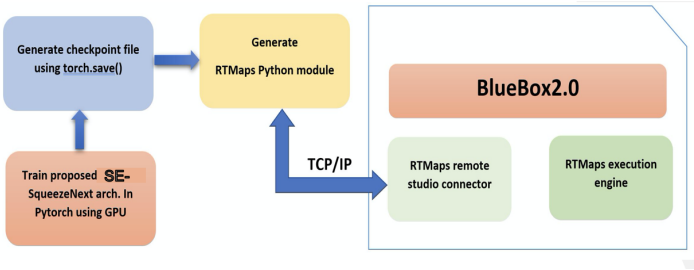



Fig. 3: Flowchart of deployment on NXP BlueBox 2.0

### B. SE-SqueezeNext on NXP i.MX RT1060 MCU

Deploying SE-SqueezeNext on NXP i.MX RT1060 involves two steps, first to convert our model to TensorflowLite model and deploying that tensorflowLite model into the board.

*1) Converting into TensorflowLite Model::* The NXP eIQ is an AI software development environment to create AI applications for implanted processors, for example, i.MX RT hybrid processors. The eIQ programming incorporates neural network compilers, and improved libraries. TensorFlow Lite is one of the derivation motors bolstered by eIQ programming with elite and enhanced memory use than TensorFlow. TFLite-Converter takes a current model in the keras system and creates the TensorFlow Lite FlatBuffer document (.tflite). The Python API for TFLiteConverter permits custom articles, for example, activation functions, loss functions and so forth to be passed during the change procedure. The IDE utilized for this change procedure is Microsoft Visual Studio Code (VS Code). This process can be visualised from Figure 4.

*2) Deploying on i.MX RT1060::* The MCU Xpresso SDK is explicitly planned by NXP to quicken application advancement in i.MX RT hybrid processors. The most recent adaptation incorporates the refreshed eIQ libraries and demos. This SDK additionally bolsters UART investigate support to run the application on Teraterm. The tflite model is changed over into a C array header file (.h) that can be imported on the board. The API call is utilized in the code to stack the model utilizing this header file. At that point, the model is fixed and we can see the result in Teraterm.

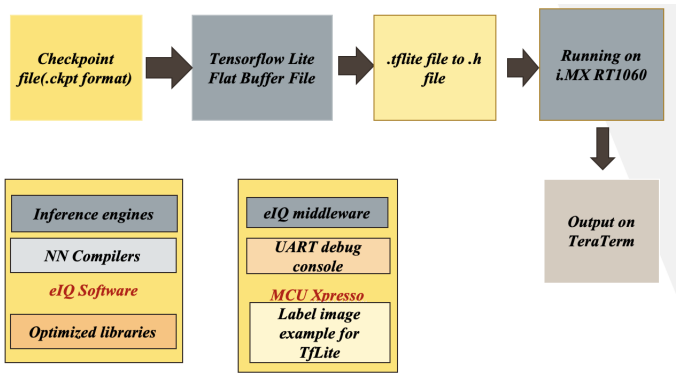



Fig. 4: Flowchart of deployment on i.MX RT1060 MCU.

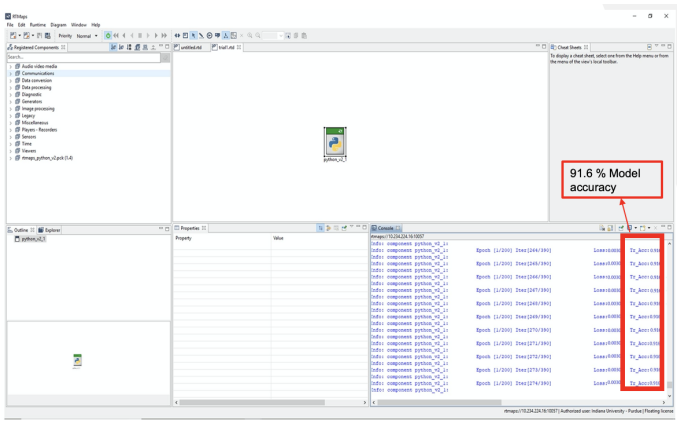



Fig. 5: The RTMaps Console Result.

## IV. Hardware and Software Requirements

- Aorus Geforce RTX 2080Ti GPU.
- Nvidia Geforce GTX 1080Ti GPU.
- Python version 3.6.7.
- Spyder version 3.6.
- Pytorch version 1.0.
- Netscope (SE-SqueezeNext visualization).
- RTMaps by Intempora
- NXP BlueBox 2.0
- NXP i.MX RT1060 MCU
- MCU Xpresso SDK
- Teraterm

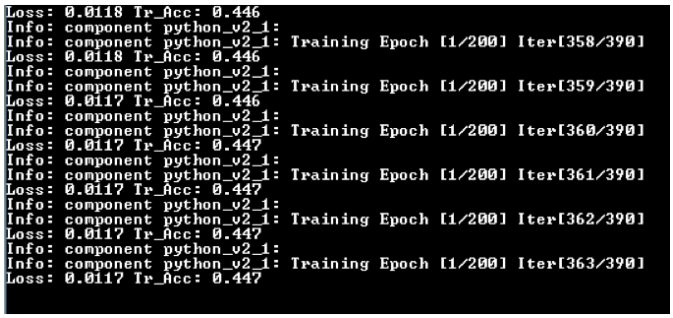



Fig. 6: The Teraterm result of the deployment.

## V. Results

In this paper, we have considered a pretrained SE-SqueezeNext model. This is trained and validated for CIFAR-10 dataset. We used Nvidia Geforce GTX 1080Ti GPU for training of the model. The original network is trained using the Pytorch framework with a total number of epochs to 200 and with a variable learning rate of 0.1, 0.01 and 0.001. We have used Stochastic gradient descent (SGD) optimizer with nestrov, decay and momentum. The batch size for training the network is 128 and for the test set, it is 64. We have replicated a similar configuration to the model with Keras as well. The results of the deployment are as follows.

### A. With NXP BlueBox 2.0

Here, we are endeavoring to make classifier work effectively on NXP Bluebox 2.0. In this way, the model is sustained with a few irregular pictures taken from the test dataset with right ground truth esteems and requesting that the model anticipate those arbitrary pictures. The RTMaps Console result is appeared underneath in Figure 5. The BlueBox result can be seen utilizing Teraterm terminal. The Teraterm result can be seen underneath in figure 6. The model is given some arbitrary information pictures like cat, boat and plane. It accurately predicts those pictures on NXP Bluebox 2.0.

### B. With NXP i.MX RT1060 MCU

Right now, attempted to give some irregular pictures like cat and plane, requesting that the model foresee them. For the most part, we gave cat and plane on the grounds that these classes have a place with the CIFAR10 dataset and our model is just prepared to this dataset. The result can be seen in Teraterm terminal. Our model is effectively capable to characterize cat and plane images effectively on NXP i.MX RT1060 alongside inference times appeared in the Figure 8.

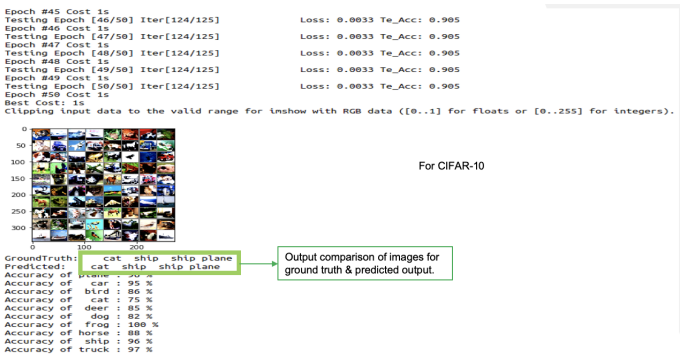
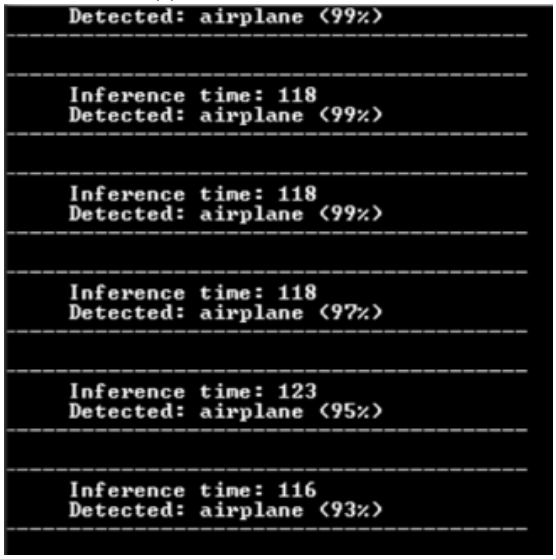



Fig. 7: The image classifier result on the console.


(a) Teraterm result of cat.




(b) Teraterm result of airplane.

Fig. 8: Results of Successful Recognition of Cat and Airplane.

## VI. Conclusion

It is evident from the results that we have successfully deployed SE-SqueezeNext architecture on flexible and highly computational embedded platforms like NXP BlueBox 2.0 and NXP i.MX RT1060. The model is as small as 0.595MB and with accuracy of 92.60% makes it the appropriate choice for the deployment on the respective embedded platforms. There can be many tweaks be implemented on the existing architecture to make it more efficient for deployment on real-time systems. This can also be further developed for object tracking and detection applications. On a further note, this model can also be tested for deployment on NXP i.MX 8M Mini MCU which is more advanced than NXP i.MX RT1060 MCU.

## References


[1] R. T. N. V. S. Chappa and M. El-Sharkawy, "Squeeze-and-Excitation SqueezeNext: An Efficient DNN for Hardware Deployment," 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2020, pp. 0691-0697.

[2] J. K. Duggal and M. El-Sharkawy, "Shallow SqueezeNext: An Efficient & Shallow DNN," 2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES), Cairo, Egypt, 2019, pp. 1-6. doi: 10.1109/ICVES.2019.8906416

[3] Duggal, Jayan Kant, 2019. Design Space Exploration of DNNs for Autonomous Systems (MSECE Thesis, Purdue University, Indianapolis).

[4] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J. and Keutzer, K., (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡ 0.5MB model size.arXiv preprint arXiv:1602.07360.

[5] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai,Xiangyu Yue, Peter Jin, Sicheng Zhao, Kurt Keutzer, (2018). SqueezeNext: Hardware-Aware Neural Network Design. arXiv preprint arXiv: 1803.10615

[6] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[7] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu, "Squeeze-and-Excitation Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2019.2913372

[8] Ashraf, Khalid, et al. "Shallow networks for high-accuracy road object-detection." arXiv preprint arXiv:1606.01561 (2016).

[9] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." arXiv preprint arXiv:1502.03167 (2015). 1

[10] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[11] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[12] B. Wu, A. Wan, X. Yue, and K. Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar p

[13] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. arXiv preprint arXiv:1612.01051, 2016.

[14] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. arXiv preprint arXiv:1611.10012, 2016. preprint arXiv:1609.07061, 2016.

[15] 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.

[16] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. "Cifar-10 (canadian institute for advanced research)." URL http://www. cs. toronto. edu/kriz/cifar. html (2010).

[17] Huang, Yanping, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." arXiv preprint arXiv:1811.06965 (2018).

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[19] T. Hsiao, Y. Chang and C. Chiu, "Filter-based Deep-Compression with Global Average Pooling for Convolutional Networks," 2018 IEEE International Workshop on Signal Processing Systems (SiPS), Cape Town, 2018, pp. 247-251.

[20] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, Quoc V (2019). Le. Learning Data Augmentation Strategies for Object Detection. arXiv preprint arXiv : 1906.11172

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. arXiv preprint arXiv:1512.00567, 2015.

[22] Ludermir, Teresa B., Akio Yamazaki, and Cleber Zanchettin. "An optimization methodology for neural network weights and architectures." IEEE Transactions on Neural Networks 17.6 (2006): 1452-1459.

[23] Chappa, Naga Venkata Sai Raviteja, 2020. Squeeze-and-Excitation SqueezeNext: An Efficient DNN for Hardware Deployment (MSECE Thesis, Purdue University, Indianapolis.)