Journal of
Low Power Electronics and Applications

MDPI

Article

CondenseNeXtV2: Light-Weight Modern Image Classifier Utilizing Self-Querying Augmentation Policies

Priyank Kalgaonkar and Mohamed El-Sharkawy *

Department of Electrical and Computer Engineering, Purdue School of Engineering and Technology, Indianapolis, IN 46254, USA; pkalgaon@purdue.edu
* Correspondence: melshark@purdue.edu

Abstract: Artificial Intelligence (AI) combines computer science and robust datasets to mimic natural intelligence demonstrated by human beings to aid in problem-solving and decision-making involving consciousness up to a certain extent. From Apple's virtual personal assistant, Siri, to Tesla's self-driving cars, research and development in the field of AI is progressing rapidly along with privacy concerns surrounding the usage and storage of user data on external servers which has further fueled the need of modern ultra-efficient AI networks and algorithms. The scope of the work presented within this paper focuses on introducing a modern image classifier which is a light-weight and ultra-efficient CNN intended to be deployed on local embedded systems, also known as edge devices, for general-purpose usage. This work is an extension of the award-winning paper entitled 'CondenseNeXt: An Ultra-Efficient Deep Neural Network for Embedded Systems' published for the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC). The proposed neural network dubbed CondenseNeXtV2 utilizes a new self-querying augmentation policy technique on the target dataset along with adaption to the latest version of PyTorch framework and activation functions resulting in improved efficiency in image classification computation and accuracy. Finally, we deploy the trained weights of CondenseNeXtV2 on NXP BlueBox which is an edge device designed to serve as a development platform for self-driving cars, and conclusions will be extrapolated accordingly.

Keywords: CondenseNeXt; convolutional neural network; computer vision; embedded systems; edge devices; image classification; CNN; PyTorch

Citation: Kalgaonkar, P.; El-Sharkawy, M. CondenseNeXtV2: Light-Weight Modern Image Classifier Utilizing Self-Querying Augmentation Policies. *J. Low Power Electron. Appl.* **2022**, *12*, 8. https://doi.org/10.3390/jlpea12010008

Academic Editor: Andrea Calimera

Received: 9 December 2021
Accepted: 21 January 2022
Published: 3 February 2022

1. Introduction

Convolutional Neural Network (CNN) is a class of Deep Neural Network (DNN), which is a subset of Machine Learning (ML), designed to realize and harness power of Artificial Intelligence (AI) to simulate natural intelligence demonstrated by living creatures of the Kingdom Animalia. The first CNN algorithm was introduced by Alexey G. Ivakhnenko and V. G. Lapa in 1967 for supervised, deep, feed-forward, multi-layer perceptions [1]. By the year 2012, data and computation intensive CNN architectures began dominating accuracy benchmarks. On 30 September 2012, AlexNet CNN architecture developed by Alex Krizhevsky in collaboration with Ilya Sutskever and Geoffrey Hinton [2,3] won the ImageNet Large Scale Visual Recognition (ILSCR) Challenge [4] which caught the world's attention, eventually becoming the benchmark for newer CNNs utilizing GPUs to accelerate deep learning [5].

In recent years, CNNs have become a popular cornerstone in the field of computer vision research and development which is a multidisciplinary field of computer science that focuses on developing innovative techniques and algorithms to enable machines to perform complex tasks such as image classification and object detection, analogous to that of a human visual system, for real-world applications such as advanced surveillance

systems for malicious activity recognition [6], self-driving cars, AI robots, and Unmanned Ariel Vehicles (UAV) such as drones [7].

In this paper, we propose a modern image classifier which is light-weight and ultra-efficient that is intended to be deployed on local embedded systems, also known as edge devices, for general-purpose usage. This work is an extension of an award-winning paper entitled 'CondenseNeXt: An Ultra-Efficient Deep Neural Network for Embedded Systems' published for the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) [8]. The proposed neural network dubbed CondenseNeXtV2 utilizes a new self-querying augmentation policy technique on a given target dataset along with adaption to the latest version of PyTorch framework at the time of writing this paper, resulting in an improved top-1 accuracy in image classification performance which we shall observe through experiments by deploying the trained weights of CondenseNeXtV2 for real-time image classification on NXP BlueBox development platform [9].

2. Background and Literature Review

2.1. Artificial Neural Networks

Artificial Neural Networks (ANN) take inspiration from the human brain. A human brain is a complex and non-linear processing system capable of retraining and reorganizing its crucial structural components known as neurons to perform complex operations such as image classification and object detection much faster than any general-purpose computer in existence today. ANNs are developed using a common programming language, trained and then deployed on computer hardware capable of executing objective-specific tasks or deployed to simulate on a computer. ANNs function in a similar way by:

1. Extracting knowledge using neurons and,
2. Storing the extracted information with the help of inter-neuron connection strengths known as synaptic weights.

This process of learning by reorganizing synaptic weights in an artificial neural network is called a learning algorithm. Figure 1 below provides a visual representation of a nonlinear model of a neuron in an ANN.

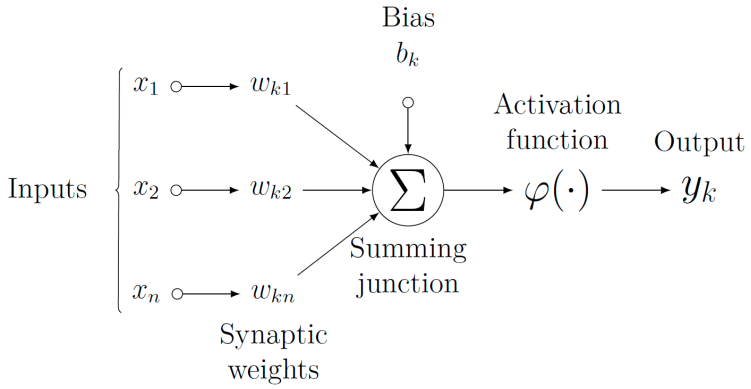

Figure 1. A visual representation of a nonlinear model of a neuron in an ANN.

In Figure 1, synaptic weights are given as inputs to the neurons of an ANN which may have positive or negative values. A summing junction functions as an adder to linearly combine all input weights with respect to corresponding synaptic weights of the neuron and a continuously differentiable linear or non-linear activation functions such as Sigmoid, Tanh, ReLU, etc. are used to decide if the neuron should be activated or not.

2.2. Representation Learning in Multi-Layer Feed-Forward Neural Networks

Multi-layer feed-forward neural networks are one of the most popular and widely used types of artificial neural network architectures in which the neurons make use of a learning algorithm to train on a target dataset.

Multi-layer feed-forward neural networks can have one or more hidden layers which determines the *depth* of a neural network. Here, the expression *hidden* signifies that a particular part of an ANN is not directly accessible to input or output nodes of the ANN. Therefore, as the depth increases, the ANN will possess even more synaptic connections between neurons to extract meaningful information and adjust its weights. Such a network architecture is also commonly known as multilayer perceptrons, which utilizes a backpropagation algorithm during the training phase.

The work proposed within this paper is based on fundamental principles of multi-layer feed-forward neural networks for supervised learning in the OpenCV realm. In supervised machine learning, the mapping between input variables (x) and output variables (y) is learned. It infers a function from labeled dataset known as ground truth. The idea is to train and learn hundreds of classes from a variety of different images containing noise and irregularities which can be a challenge for traditional image classification algorithms. Figure 2 below provides a visual representation of these images. The work presented within this paper utilizes AutoAugment [10] data augmentation technique to overcome this issue as discussed in the following sections of this paper.

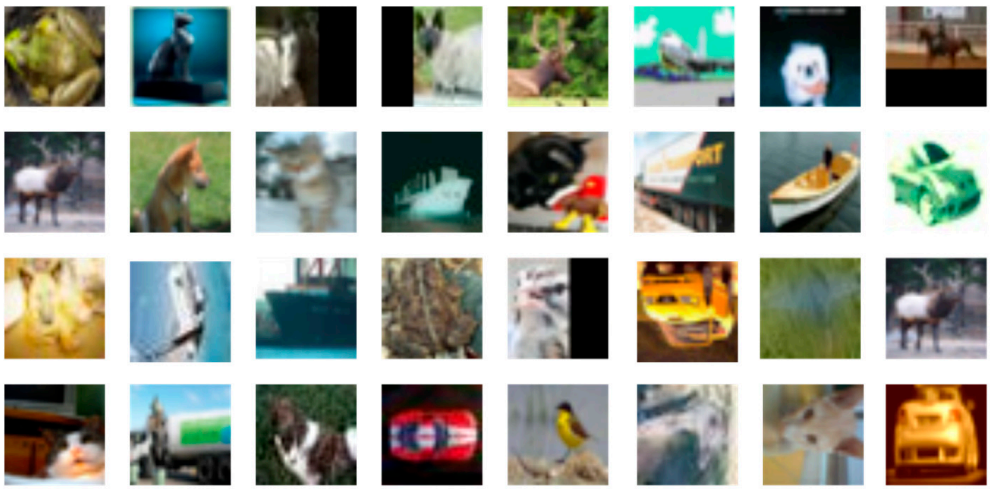

Figure 2. Sample of 32 × 32 resolution images containing noise and other irregularities.

To address the challenges of performance and efficiency of deep neural network architectures, in 2016, a team of researchers and engineers of Facebook AI Research (FAIR) lab introduced PyTorch and released it under the Modified BSD license for fair use [11]. PyTorch is a scientific computing framework and an open-source machine learning library built upon the Torch library, which is also a scientific computing framework and an open-source machine learning library, introduced in 2002 [12]. Some of the important features of PyTorch include:

1. Tensor computation accelerated by GPU(s).
2. Automatic differentiation using tape-based autograd.

PyTorch is a popular choice of deep learning framework for commercial applications such as autonomous vehicles, AI applications, etc. Tesla Motors, a leader in the autonomous (self-driving) vehicle industry, utilizes PyTorch for their famous Autopilot system, which is a suite of SAE Level 2 Advanced Driver-Assistance System (ADAS) features, by utilizing advanced computing hardware for deep learning, computer vision and sensor fusion [13]. The proposed CNN within this paper is based on the latest stable release of PyTorch version 1.10.0 and CUDA toolkit version 10.2 [14].

2.3. Evolution of CondenseNeXt

Huang et al. introduced ResNeXt [15] CNN architecture which utilizes an innovative technique to skip one or more layers, known as identity shortcut connections. This allowed ResNeXt CNN to train on large amounts of data with unparalleled efficiency at the time.

In the following year, in 2017, Huang et al. introduced DenseNet [16] CNN architecture that utilizes a novel technique to connect each convolutional layer to other proceeding layers in a feed-forward fashion. This facilitates the reuse of synaptic weights by providing outputs of current layer as inputs to every other proceeding layer resulting in extraction of information at different levels of coarseness. DenseNet provides a baseline for CondenseNet CNN architecture.

In 2018, Huang et al. introduced CondenseNet [17], which was an improvement over the DenseNet CNN. In this CNN architecture, authors proposed pruning several layers and blocks and replacing standard convolutions with group convolutions (G). Furthermore, they proposed a technique to learn mapping patterns between channels and groups by connecting layers from different groups to each other directly, thus preventing individual groups of channels from training separately. The authors collectively called this technique as learned group convolutions. Figure 3 below provides a visual comparison of dense blocks in DenseNet vs. CondenseNet CNN.

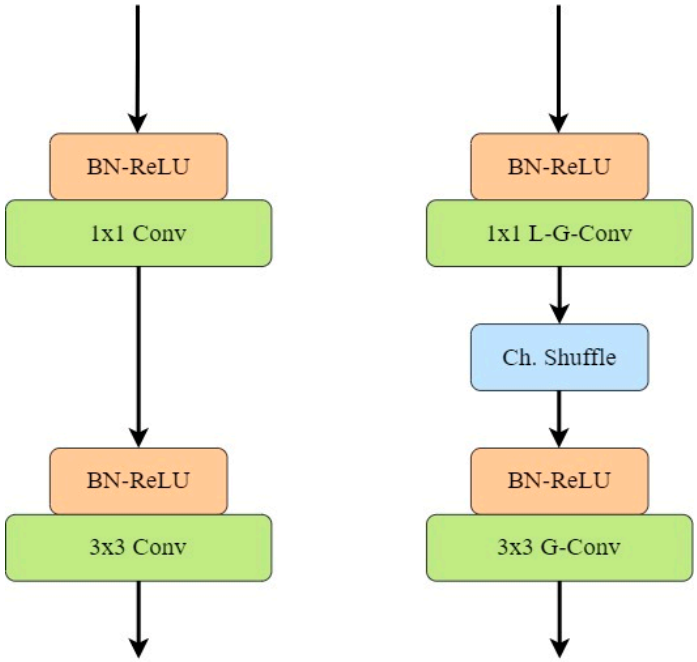

Figure 3. Comparison of dense blocks in DenseNet vs. CondenseNet CNN.

CondenseNet performs a 1×1 standard group convolution. The major drawback of implementing such a convolution is that it does not parse through each channel's spatial dimensions and therefore, misses out on fine details leading to a loss in efficiency and overall accuracy of the network. To address these issues, Kalgaonkar and El-Sharkawy introduced CondenseNeXt [8] in 2021. CondenseNeXt refers to the *next* dimension of cardinality. In this CNN architecture, the authors propose replacing standard grouped convolutions with depthwise separable convolutions [18], which allows the network to extract information by transforming an image only once and then elongating this transformed image to n number of channels instead of simply transforming the image n number of times, resulting in a boost in computational efficiency and fewer number of Floating-Point Operations (FLOPs) at the training stage.

3. CondenseNeXtV2 Architecture

In this section, a new CNN architecture, called CondenseNeXtV2, is being proposed. CondenseNeXtV2 is inspired by and is an improvement over CondenseNeXt CNN. The primary goal of CondenseNeXtV2 CNN is to further improve performance and top-1 accuracy of the network. This section describes the architecture in detail.

3.1. Convolutional Layers

CondenseNeXtV2 utilizes depthwise separable convolution which enhances the efficiency of the network without having to transform an image over and over again, resulting in reduction in number of FLOPs and an increase in computational efficiency in comparison to standard group convolutions. Depthwise separable convolution is comprised of two distinct layers:

- Depthwise convolution: This layer acts like a filtering layer which applies depthwise convolution to a single input channel. Consider an image of size $X \times X \times I$ given as an input with I number of input channels of the input image with kernels (K) of size $K \times K \times 1$, the output of this depthwise convolution operation will be $Y \times Y \times I$ resulting in reduction of spatial dimensions whilst preserving I number of channels (depth) of the input image. The cost of computation for this operation can be mathematically represented as follows:

$$\hat{Y}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} * X_{k+i-1,l+j-1,m} \tag{1}$$

- Pointwise convolution: This layer acts like a combining layer which performs linear combination of all outputs generated by the previous layer since the depth of input image from previous operation hasn't changed. Pointwise convolution operation is performed using a 1×1 kernel i.e., $1 \times 1 \times K$ resulting in the size of the output image to be $Y \times Y \times K$. The cost of computation for this operation can be mathematically represented as follows:

$$Y_{k,l,n} = \sum_{m} \widetilde{K}_{m,n} * \hat{Y}_{k-1,l-1,m} \tag{2}$$

A depthwise separable convolution splits a kernel into two distinct layers for filtering and combining operations resulting in an improvement in overall computational efficiency. Figure 4 provides a 3D visual representation of depthwise separable convolution operation where a single image is transformed only once and then this transformed image is elongated to over 128 channels. This allows the CNN to extract information at different levels for coarseness. For example, consider a green channel from the RGB channel being processed. Depthwise separable convolution will extract different shades of color green from this channel.

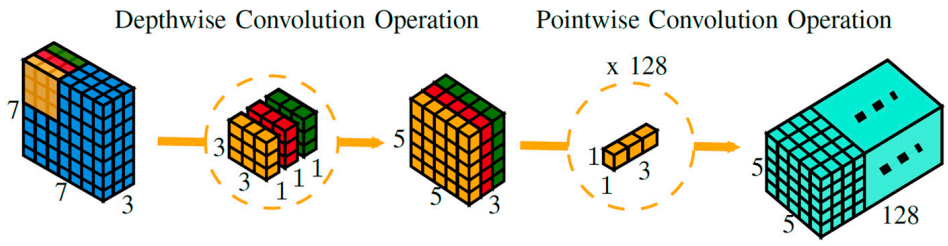

Figure 4. A 3D visual representation of depthwise separable convolution for a given example.

3.2. Model Compression

3.2.1. Network Pruning

Network pruning is one of the effective ways to reduce computational costs by discarding redundant and insignificant weights that do not affect the overall performance of a DNN. Figure 5 provides a visual comparison of weight matrices for different pruning approaches as follows:

1. Fine-grained pruning: using this approach, redundant weights that will have minimum influence on accuracy will be pruned. However, this approach results in irregular structures.
2. Coarse-grained pruning: using this approach, an entire filter which will have minimal influence on overall accuracy is discarded. This approach results in regular structures. However, if the network is highly compressed, there may be significant loss in the overall accuracy of the DNN.
3. 2D group-level pruning: this approach maintains a balance of both, benefits and trade-offs of fine-grained and coarse-grained pruning.

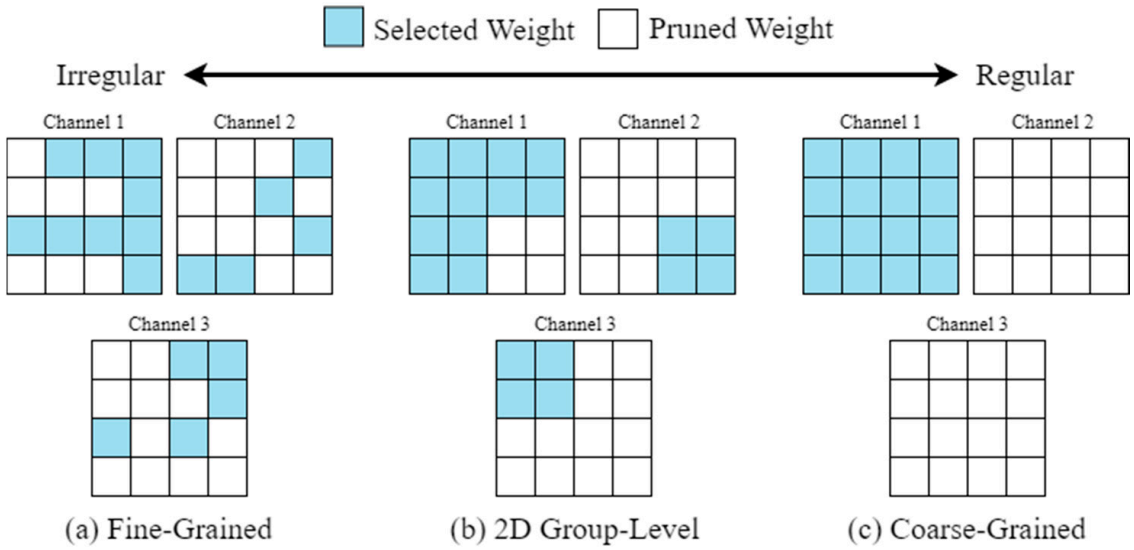

Figure 5. Comparison of different pruning methods.

CondenseNeXtV2 CNN utilizes 2D Group-Level pruning to discard insignificant filters which is complemented by L_1-Normalization [19] to facilitate the group-level pruning process.

3.2.2. Class-Balanced Focal Loss (CFL) Function

Pruning networks in order to discard insignificant and redundant elements of the network can still have an overall harsh effect caused due to imbalanced weights. Therefore, in order to mitigate any such issues, CondenseNeXtV2 CNN incorporates a weighting factor inversely proportional to the total number of samples, called Class-balanced Focal Loss (CFL) Function [20]. This approach can be mathematically represented as follows:

$$CFL(z, y) = -\sum_{i=1}^{C} (1 - p_i^t)^{\gamma} * \log(p_i^t) \tag{3}$$

3.2.3. Cardinality

A new dimension, called Cardinality, is added to the existing spatial dimensions of the CondenseNeXtV2 network to prevent loss in overall accuracy during the pruning process, denoted by D. Increasing cardinality rate is a more effective way to obtain a

boost in accuracy instead of going wider or deeper within the network. Cardinality can be mathematically represented as follows:

$$G * G_x = X * D - p \cdot X \tag{4}$$

3.3. Data Augmentation

Data augmentation is a popular technique to increase the amount of data and its diversity by modifying already existing data and adding newly created data to existing data in order to improve the accuracy of modern image classifiers. CondenseNeXtV2 utilizes AutoAugment data augmentation technique to pre-process target dataset to determine the best augmentation policy with the help of Reinforcement Learning (RL). AutoAugment primarily consists of following two parts:

- Search Space: Each policy consists of five sub-policies and each sub-policy consists of two operations that can be performed on an image in a sequential order. Furthermore, each image transformation operation consists of two distinct parameters as follows:
    - The probability of applying the image transformation operation to an image.
    - The magnitude with which the image transformation operation should be applied to an image.

There are a total of 16 image transformation operations. 14 operations belong to the PIL (Python Imaging Library) which provides the python interpreter with image editing capabilities such as rotating, solarizing, color inverting, etc. and the remaining two operations are Cutout [21] and SamplePairing [22]. Table 6 in the 'Supplementary materials' section of [10] provides a comprehensive list of all 16 image transformation operations along with default range of magnitudes.

- Search Algorithm: AutoAugment utilizes a controller RNN, a one-layer LSTM [23] containing 100 hidden units and 2×5B softmax predictions, to determine the best augmentation policy. It examines the generalization ability of a policy by performing *child model* (a neural network being trained during the search process) experiments on a subset of the corresponding dataset. Upon completion of these experiments, a reward signal is sent to the controller to update validation accuracy using a popular policy gradient method called Proximal Policy Optimization algorithm (PPO) [24] with a learning rate of 0.00035 and an entropy penalty with a weight of 0.00001. The controller RNN provides a decision in terms of a softmax function which is then fed into the controller's next stage as an input, thus allowing it to determine magnitude and probability of the image transformation operation.

At the end of this search process, sub-policies from top five best policies are merged into one best augmentation policy with 25 sub-policies for the target dataset.

3.4. Activation Function

An activation function in a neural network plays a crucial role in determining the output of a node by limiting the output's amplitude. These functions are also known as transfer functions. Activation functions help the neural network learn complex patterns of the input data more efficiently whilst requiring fewer number of neurons.

CondenseNeXtV2 CNN utilizes ReLU6 activation functions along with batch normalization before each layer. ReLU6 activation function cap units at 6 which helps the neural network learn sparse features quickly in addition to preventing gradients to increase infinitely as seen in Figure 6. ReLU6 activation function can be mathematically defined as follows:

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

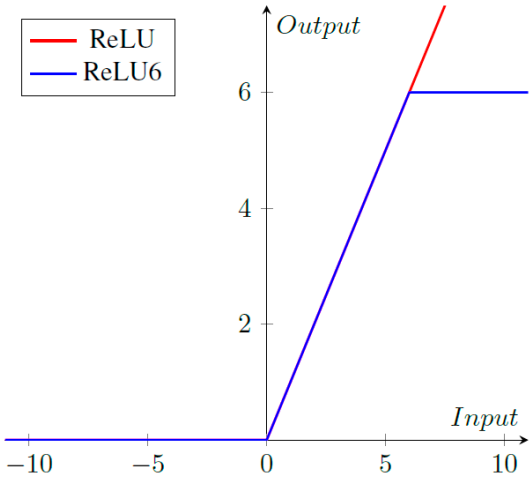

Figure 6. ReLU vs. ReLU6 activation functions.

4. NXP BlueBox 2.0 Embedded Development Platform

This section provides a brief introduction of NXP BlueBox 2.0 (Gen2) embedded development platform for Automotive High-Performance Compute (AHPC). It is a series of development platform designed for self-driving (autonomous) vehicles.

4.1. System Design of NXP BlueBox Gen2 Family

The BlueBox 2.0 embedded development platform series is designed and developed by NXP Semiconductors. First launched in 2016 as a generation 1 series, the target application has been self-driving (autonomous) vehicles ever since. It was introduced to general public in Austin, Texas, USA. Shortly thereafter, NXP improved an advanced series of BlueBox which was called BlueBox 2.0, the second generation of this series. It features following three new ARM-based automotive processors [9]:

1. S32V234 (S32V) dedicated for computer vision processing tasks: This automotive processor is based on a four core ARM Cortex A53 architecture and operates at 1.0 GHz. An additional ARM Cortex M4 CPU provides further support and functionality for reliability and workload sharing. It is complimented by 4 MB SRAM and an additional 32-bit LPDDR3 controller to support additional memory if required by the developers. It also boasts an Image Signal Processor (ISP) on-board the SoC for noise filtering, noise reduction, etc.
2. LS2084A (LS2) dedicated for high performance computing tasks: This automotive processor is based on an eight core ARM Cortex A72 architecture and operates at 1.80 GHz. It is complimented by 144 bytes of DDR4 RAM with support enabled for LayerScape LX2 family. NXP claims LS2 to provide 15 years of reliable service and meets Q100 grade 3 automotive standards.
3. S32R274 (S32R) dedicated for radar information processing tasks: This automotive processor is based on a dual-core Freescale PowerPC e200z4 architecture and operates at 120 MHz in addition to a checker core on-board this System-on-Chip (SoC). It features a 2.0 MB flash memory and 1.5 MB of SRAM.

NXP BlueBox 2.0 adheres to automotive compliance and safety standards such as the ASIL-B/C and D to ensure operability and reliability in all real-world conditions. Figure 7 below provides a high-level system architecture view of NXP BlueBox 2.0 [9].

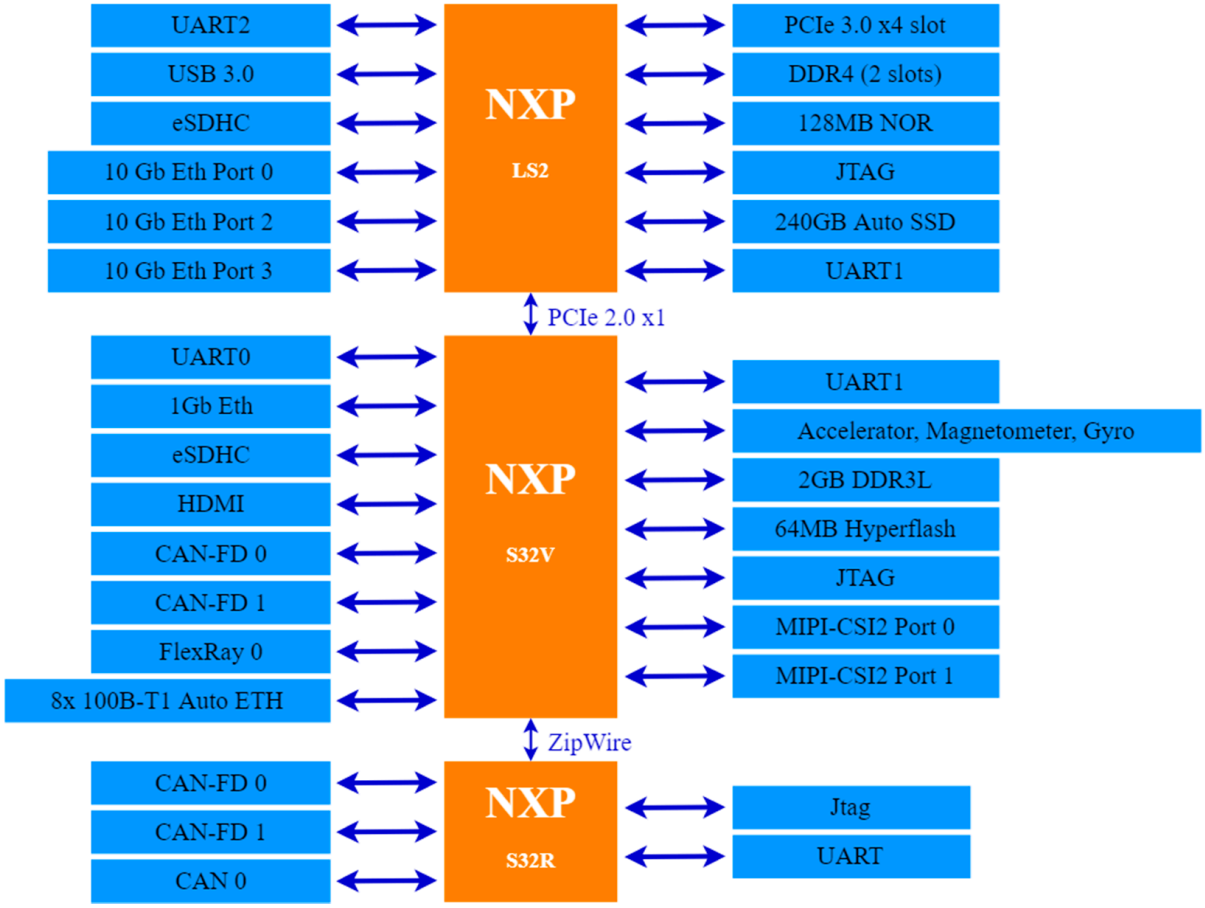

Figure 7. A high-level system architecture view of NXP BlueBox 2.0.

Figure 8 below provides a top-frontal view NXP BlueBox 2.0 automotive embedded development platform.

Figure 8. NXP BlueBox 2.0.

4.2. RTMaps (Real-Time Multi-Sensor Applications) Remote Studio Software

RTMaps (Real-time Multi-sensor Applications) Remote Studio Software is a component-based software development and execution environment made by Intempora to facilitate development of autonomous driving and related automotive applications [25]. It allows users to link various sensors and hardware components by providing a host of component libraries for automotive sensors, buses, and perception algorithm design as well as support for almost any type of sensors and any number of those sensors.

RTMaps can run on Windows and Linux operating systems and provides support for PyTorch and TensorFlow deep learning frameworks. In RTMaps, programming is done using Python scripting language and then deployed for real-time inference on NXP BlueBox 2.0.

5. Cyberinfrastructure

This section provides details on the hardware and software utilized for training and testing the proposed CondenseNeXtV2 CNN architecture from which results are extrapolated accordingly in the following sections.

5.1. Cyberinfrastructure for Training the Proposed CNN

Cyberinfrastructure for training is provided and managed by the Research Technologies division at the Indiana University in part by Shared University Research grants from IBM Inc. to Indiana University and Lilly Endowment Inc. through its support for the Indiana University Pervasive Technology Institute [26]. It is as follows:

- Intel Xeon Gold 6126 12-core CPU with 32 GB RAM.
- NVIDIA Tesla V100 GPU.
- CUDA Toolkit 10.2.
- Python version 3.7.9.
- PyTorch version 1.10.

5.2. Cyberinfrastructure for Testing the Proposed CNN

Cyberinfrastructure for testing is provided and managed by the Internet of Things (IoT) Collaboratory at the Purdue School of Engineering and Technology at Indiana University Purdue University at Indianapolis as follows:

- NXP BlueBox 2.0 embedded development platform.
- Intempora RTMaps Remote Studio version 4.9.0.
- Python version 3.6.7.
- PyTorch version 1.10.

6. Experiments and Results

This section provides information on experiment results based on extensive training and testing of the proposed convolutional neural network, CondenseNeXtV2, on CIFAR-10 and CIFAR-100 benchmarking datasets to verify real-time image classification performance on NXP BlueBox 2.0. CondenseNeXtV2 CNN has been adapted to be compatible with the latest stable release of PyTorch version 1.10.0 and CUDA toolkit version 10.2 [14] whereas CondenseNet and CondenseNeXt CNNs are based on PyTorch version 1.1.0 and 1.9.0 respectively.

6.1. CIFAR-10 Dataset

CIFAR-10 dataset is a popular computer-vision dataset used for image classification and object detection. It consists of 60,000 32×32 RGB color images equally divided into 10 object classes with 6,000 images per class. CIFAR-10 dataset is split into two sets: 50,000 images for training a CNN and 10,000 images for testing the performance of trained model weights. The training and testing sets are mutually exclusive to obtain fair evaluation results.

CondenseNeXtV2 CNN was trained for 200 epochs with a batch size of 64 input images. Since learning algorithm for CondenseNet, CondenseNeXt and CondenseNeXtV2 is unique, hyperparameters such as growth and learning rate for training these CNNs is different. Python scripting language was utilized to develop an image classification algorithm for CIFAR-10 dataset and the trained weights were then deployed on to NXP BlueBox 2.0 embedded development platform using RTMaps Remote Studio software for image classification performance evaluation.

Table 1 provides an overview of testing results in terms of FLOPs, number of trainable parameters, Top-1 accuracy and inference time when evaluated on NXP BlueBox 2.0. Top-1 accuracy corresponds to one class predicted by CNN with highest probability of matching the expected answer, also known as the ground truth. Figure 9 provides an overview of model performance curves and Figure 10 provides screenshot of single image classification performance of CondenseNeXtV2 when evaluated on NXP BlueBox 2.0 using RTMaps Remote Studio software on a Linux PC.

Table 1. Comparison of CIFAR-10 single image classification performance.

CNN	FLOPs (in Million)	Parameters (in Million)	Top-1 Accuracy	Inference Time (in Seconds)
CondenseNet	65.81	0.52	94.69%	0.1346
CondenseNeXt	23.80	0.16	92.28%	0.0659
CondenseNeXtV2	23.80	0.16	93.57%	0.0528

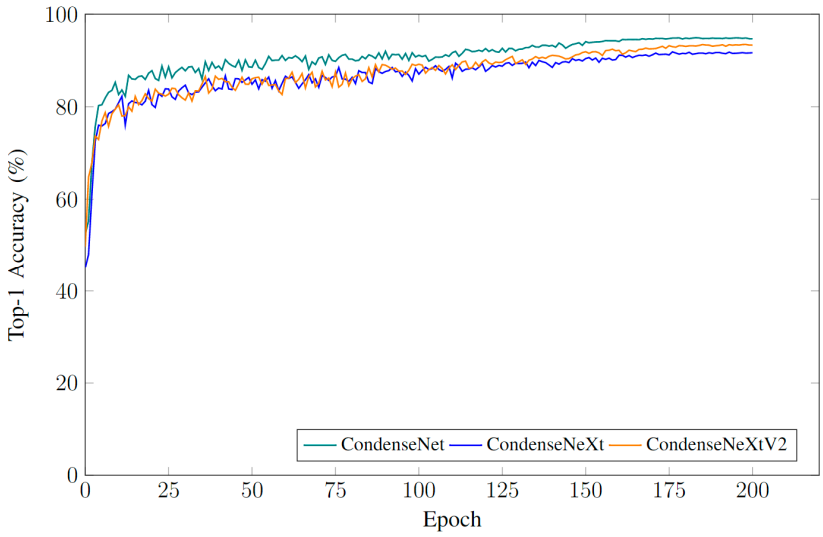

Figure 9. Performance curves of CondenseNeXtV2 vs. CondenseNet and CondenseNeXt. These CNN models are trained on CIFAR-10 dataset for 200 epochs each.

6.2. CIFAR-100 Dataset

CIFAR-100 dataset is another popular computer-vision dataset used for image classification and object detection. It also consists of 60,000 32 × 32 RGB color images equally divided into 100 object classes with 600 images per class. CIFAR-100 dataset is split into two sets: 50,000 images for training a CNN and 10,000 images for testing the performance of trained model weights. As with CIFAR-10 dataset, the training and testing sets are also mutually exclusive in this dataset to obtain fair evaluation results.

CondenseNeXtV2 CNN was trained for 240 epochs with a batch size of 64 input images. Hyperparameters such as growth and learning rate for training these CNNs is different because learning algorithm for CondenseNet, CondenseNeXt and CondenseNeXtV2 is unique. Python scripting language was utilized to develop an image classification algorithm for CIFAR-100 dataset and the trained weights were then deployed on to NXP BlueBox

2.0 embedded development platform using RTMaps Remote Studio software for image classification performance evaluation.

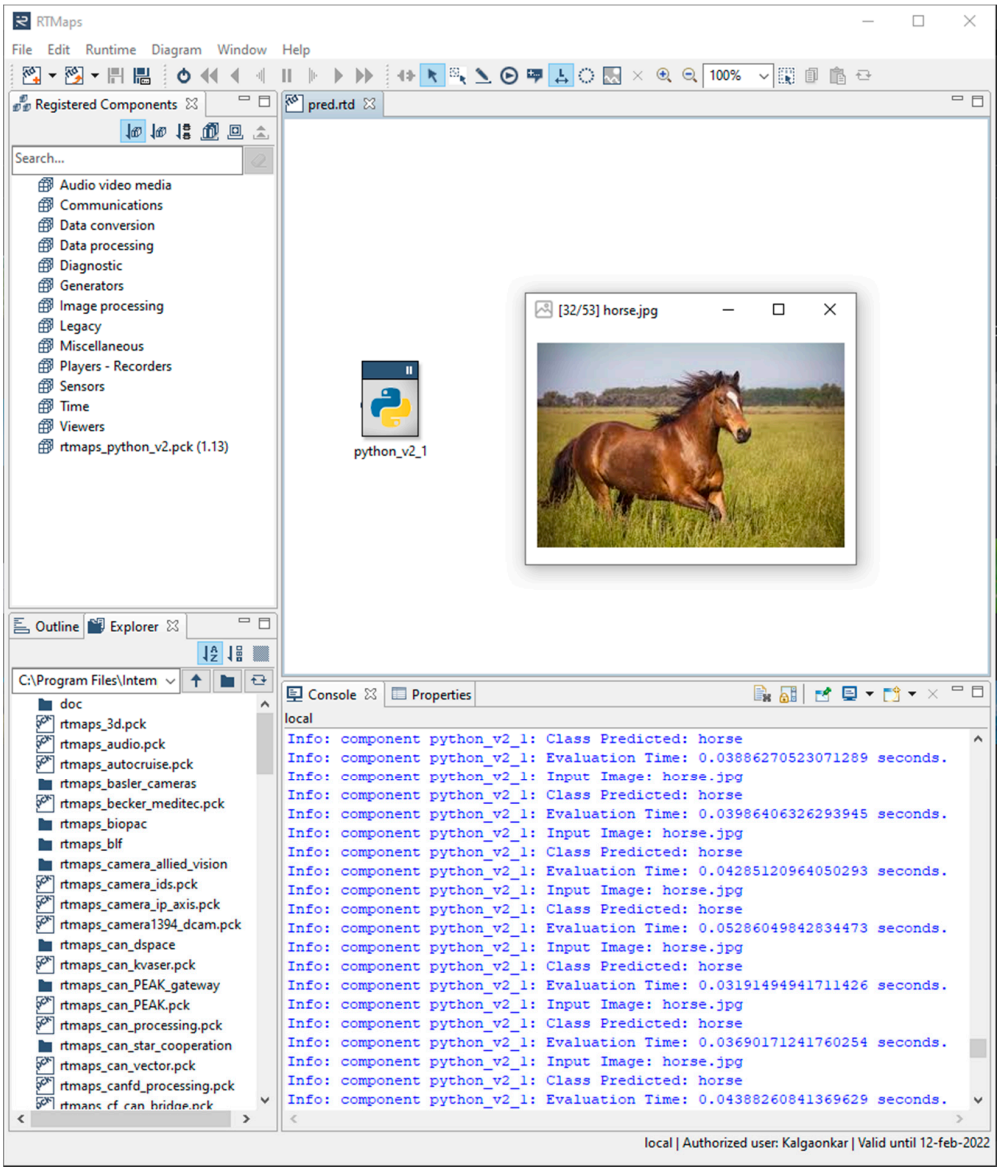

Figure 10. NXP BlueBox 2.0 single image classification prediction of a horse as an input image being observed in RTMaps Remote Studio software.

Table 2 provides an overview of testing results in terms of FLOPs, number of trainable parameters, Top-1 accuracy and inference time when evaluated on NXP BlueBox 2.0. Figure 11 provides an overview of model performance curves and Figure 12 provides screenshot of single image classification performance of CondenseNeXtV2 when evaluated on NXP BlueBox 2.0 using RTMaps Remote Studio software on a Linux PC.

Table 2. Comparison of CIFAR-100 single image classification performance.

CNN	FLOPs (in Million)	Parameters (in Million)	Top-1 Accuracy	Inference Time (in Seconds)
CondenseNet	65.85	0.55	76.65%	0.2483
CondenseNeXt	26.38	0.22	74.87%	0.1125
CondenseNeXtV2	26.38	0.22	75.54%	0.0972

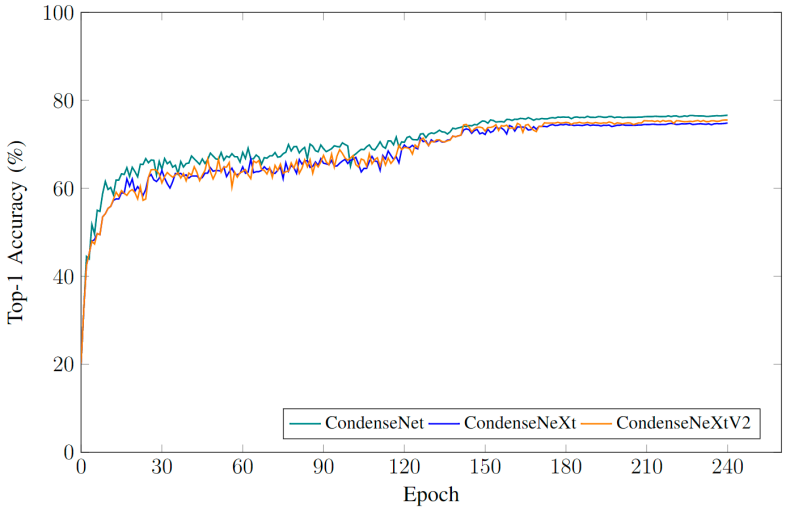

Figure 11. Performance curves of CondenseNeXtV2 vs. CondenseNet and CondenseNeXt. These CNN models are trained on CIFAR-100 dataset for 240 epochs each.

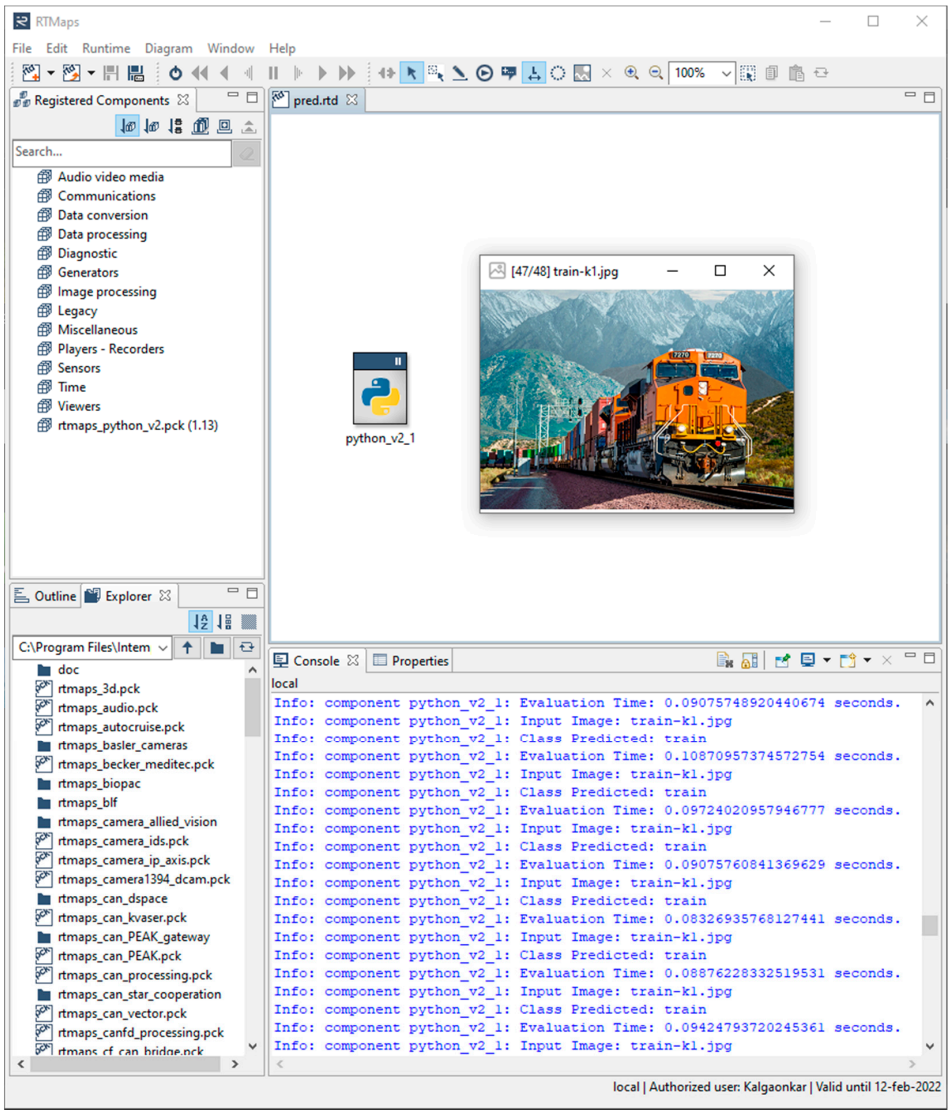

Figure 12. NXP BlueBox 2.0 single image classification prediction of a train as an input image being observed in RTMaps Remote Studio software.

7. Conclusions

The scope of the work presented within this paper focuses on introducing a modern image classifier named CondenseNeXtV2 which is light-weight and ultra-efficient that is intended to be deployed on local embedded systems, also known as edge devices, for general-purpose usage. This proposed CNN utilizes a new self-querying augmentation policy technique on a target dataset along with adaption to the latest version of PyTorch framework and activation functions resulting in improved efficiency in image classification computation and accuracy.

CondenseNeXtV2 CNN was extensively trained from scratch and tested on two benchmarking datasets: CIFAR-10 and CIFAR-100 in order to verify the single image classification performance. The trained weights of CondenseNeXtV2 were then deployed on NXP BlueBox which is an edge device designed to serve as a development platform for self-driving cars, and the results were extrapolated and compared to its baseline architecture, CondenseNet, accordingly. CondenseNeXtV2 demonstrates an increase in Top-1 accuracy over CondenseNeXt as well as utilizes fewer number of FLOPs and total number of trainable parameters in comparison to CondenseNet, observed during the training and testing phases.

Author Contributions: Conceptualization, P.K. and M.E.-S.; methodology, P.K.; software, P.K.; validation, P.K. and M.E.-S.; writing—original draft preparation, P.K.; writing—review and editing, P.K. and M.E.-S.; supervision, M.E.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to the fact that all the authors work at the same institution and there is no necessity to create repository for data exchange.

Acknowledgments: The authors like to acknowledge the Indiana University Pervasive Technology Institute for providing supercomputing and storage resources as well as the Internet of Things (IoT) Collaboratory at the Purdue School of Engineering and Technology at Indiana University Purdue University at Indianapolis for providing autonomous vehicle development platform that have contributed to the research results reported within this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berners-Lee, C.M. Cybernetics and Forecasting. *Nature* **1968**, *219*, 202–203. [CrossRef]
2. Gershgorn, D. ImageNet: The Data that Spawned the Current AI Boom, QUARTZ. Available online: https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/ (accessed on 10 January 2022).
3. Alex, K.; Sutskever, I.; Geoffrey, H. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
4. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
5. Deshpande, A. Engineering at Forward | UCLA CS '19. Available online: https://adeshpande3.github.io/ (accessed on 10 January 2022).
6. Dimitriou, N.; Kioumourtzis, G.; Sideris, A.; Stavropoulos, G.; Taka, E.; Zotos, N.; Leventakis, G.; Tzovaras, D. An integrated framework for the timely detection of petty crimes. In Proceedings of the 2017 European Intelligence and Security Informatics Conference, Athens, Greece, 11–13 September 2017. [CrossRef]
7. Lotfi, A.; Bouchachia, H.; Gegov, A.; Langensiepen, C.; McGinnity, M. (Eds.) *Advances in Computational Intelligence Systems*; Springer: Berlin/Heidelberg, Germany, 2018; p. 840. [CrossRef]
8. Kalgaonkar, P.; El-Sharkawy, M. CondenseNeXt: An Ultra-Efficient Deep Neural Network for Embedded Systems. In Proceedings of the 2021 IEEE 11th Annual Computing and Communication Workshop and Conference, CCWC 2021, Las Vegas, NV, USA, 27–30 January 2021; pp. 524–528. [CrossRef]
9. Cureton, C.; Douglas, M. *Bluebox Deep Dive—NXP's AD Processing Platform*; NXP: Einhofen, The Netherlands, 2019; p. 28.
10. Cubuk, E.D.; Zoph, B.; Mane, D.; Vasudevan, V.; Le, Q.V. AutoAugment: Learning Augmentation Policies from Data. *Cvpr* **2019**, *2018*, 113–123.

11. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Processing Syst.* **2019**, *32*, 8026–8037.
12. Collobert, R.; Kavukcuoglu, K.; Farabet, C. Torch7: A Matlab-like Environment for Machine Learning. *NIPS*. 2011. Available online: https://infoscience.epfl.ch/record/192376/files/Collobert_NIPSWORKSHOP_2011.pdf (accessed on 10 January 2022).
13. Fridman, L.; Ing, L.D.; Jenik, B.; Reimer, B. Arguing Machines: Human Supervision of Black Box AI Systems That Make Life-Critical Decisions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Long Beach, CA, USA, 16–20 June 2019. [CrossRef]
14. PyTorch. PyTorch 1.10 Release, including CUDA Graphs APIs, Frontend and Compiler Improvements. Available online: https://pytorch.org/blog/pytorch-1.10-released/ (accessed on 10 January 2022).
15. Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 1492–1500. [CrossRef]
16. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708. [CrossRef]
17. Huang, G.; Liu, S.; van der Maaten, L.; Weinberger, K.Q. CondenseNet: An Efficient DenseNet using Learned Group Convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 2752–2761. [CrossRef]
18. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258. [CrossRef]
19. Wu, S.; Li, G.; Deng, L.; Liu, L.; Wu, D.; Xie, Y.; Shi, L. L1-Norm Batch Normalization for Efficient Training of Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2043–2051. [CrossRef] [PubMed]
20. Cui, Y.; Jia, M.; Lin, T.Y.; Song, Y.; Belongie, S. Class-balanced loss based on effective number of samples. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 9268–9277. [CrossRef]
21. DeVries, T.; Taylor, G.W. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv* **2017**, arXiv:1708.04552.
22. Inoue, H. Data Augmentation by Pairing Samples for Images Classification. *arXiv* **2018**, arXiv:1801.02929.
23. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
24. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Openai, O.K. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
25. IUPUI. Speed Is Key to Safety—dSPACE. Available online: https://www.dspace.com/en/inc/home/applicationfields/stories/iupui-speed-is-key-to-safety.cfm (accessed on 10 January 2022).
26. Stewart, C.A.; Welch, V.; Plale, B.; Fox, G.; Pierce, M.; Sterling, T. Indiana University Pervasive Technology Institute. IUScholarWorks. Available online: https://scholarworks.iu.edu/dspace/handle/2022/21675 (accessed on 10 January 2022).