Sensor-Guided Adaptive Machine Learning on Resource-Constrained Devices

Franz Papst
papst@tugraz.at
Graz University of Technology / Complexity Science Hub
Graz / Vienna, Austria

Daniel Kraus
daniel.kraus@pro2future.at
Pro2Future GmbH/ Graz University of Technology
Graz, Austria

Martin Rechberger
mrechberger@student.tugraz.at
Graz University of Technology
Graz, Austria

Olga Saukh
saukh@tugraz.at
Graz University of Technology / Complexity Science Hub
Graz / Vienna, Austria

Abstract

In recent years, the deployment of deep learning models has extended beyond typical cloud environments to resource-constrained devices such as edge devices and smartphones. This shift is driven by their success in learning and detecting patterns in data. However, deep models are often excessively large and lack robustness to minor input transformations. To solve the challenge, deep learning models are often trained with data augmentation, which requires an even larger model to accommodate the additional knowledge. In this paper, we study ways to mitigate these problems by leveraging additional sensing modalities to a) adapt the input data and b) adapt the model for typical transformations. We show that both approaches increase the accuracy of deep learning models by up to 6.21% and 7.57% respectively, while using roughly the same number of parameters or even less at inference time. We furthermore study how well these approaches can handle noisy sensor readings.

CCS Concepts

• **Computer systems organization** → Sensor networks; • **Computing methodologies** → **Neural networks**; *Learning settings.*

Keywords

Machine Learning, Sensor Guided, Data Augmentation, Model Adaptation, Resource Efficient

ACM Reference Format:
Franz Papst, Daniel Kraus, Martin Rechberger, and Olga Saukh. 2024. Sensor-Guided Adaptive Machine Learning on Resource-Constrained Devices. In *14th International Conference on the Internet of Things (IoT 2024), November 19–22, 2024, Oulu, Finland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3703790.3703801

1 Introduction

The largest contributing factors for the recent success of machine learning (ML) models are the number of model parameters, the size of the dataset and the amount of compute [19]. Hence, deep learning models demonstrate optimal performance in cloud environments, where abundant computational resources are readily available. However, using a cloud is not possible or advisable for all domains, *e.g.*, due to privacy concerns or due to connectivity issues. But those domains would still benefit from the data processing capabilities of deep learning models.

Furthermore, deep learning models are often trained on datasets consisting of well-curated canonical images, but at deployment the model might be confronted with data outside the training distribution [3]. Consider a camera at the end of a production line, mounted to monitor the output of the production line. The camera feed is used as input for an ML model to detect and count the output of the production line. Due to construction constraints, the camera has to be installed in such a way, where it is tilted by a certain angle. The model was trained without any rotated images. This leads to a lower accuracy of the ML model and generally lower usefulness of the system, as it often miscounts the output. One way to tackle this issue is by training the model with rotated images. If the rotation angle is known in advance, this is trivial. However, if the angle is unknown and the model must handle any rotation, data augmentation is used. Randomly transformed images, *e.g.*, rotations, are added to the training set to teach the model those transformations. Data augmentation (DA) requires models with enough capacity to learn from the increased number of samples, which is not always desirable.

There is a vast body of literature on data augmentations [1, 3, 5, 6, 17, 18, 21, 28, 31, 32, 39, 43], but there are other approaches to solve the issue of input transformation. Invariant architectures are among the most popular and well-studied among those approaches. Convolutional Neural Networks (CNNs) [41] are commonly believed to be translation invariant[1] and can also be made invariant to rotation and scale [38]. While CNNs possess numerous advantageous properties [16], they remain fragile and could benefit from enhanced robustness [35].

Embedded devices such as smartphones often have additional sensing modalities, *e.g.*, IMU sensors, to measure metrics like the device's rotation. This auxiliary information can serve as an extra source for the ML model, enhancing its robustness to transformations without the need for additional storage for all data augmentations. This work studies how sensor-guided adaptive machine learning can improve model accuracy and how well it works compared to

[1] Although there is newer work, which argues that they are not [2].

IoT 2024, November 19–22, 2024, Oulu, Finland

ACM ISBN 979-8-4007-1285-2/24/11
https://doi.org/10.1145/3703790.3703801

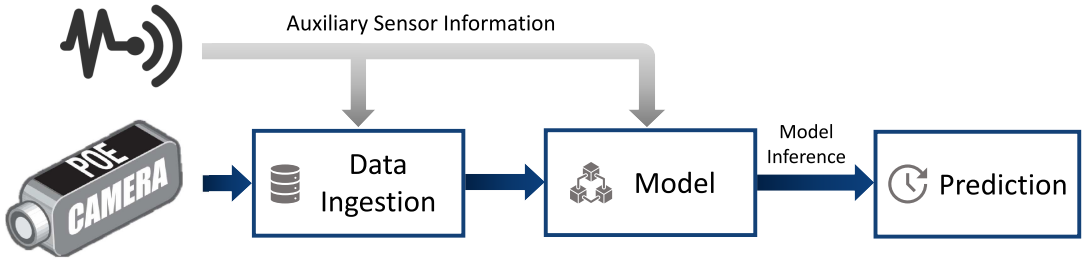


Figure 1: Schematic of a typical data ingestion pipeline for a machine learning model used to classify images from a camera.

simply using data augmentation. Using sensor information to guide a data acquisition and processing system also happens outside of machine learning, *e.g.*, the James Webb Space Telescope utilizes measurements from a dedicated fine guiding sensor to achieve its intended image quality for the Near Infrared Imager and Slitless Spectrograph [25]. Since sensor data in an IoT context is often noisy, we also investigate how well the different methods work when the sensor signal is noisy.

We study six methods: 1) models trained on canonical data, 2) models trained with augmentation, 3) larger models trained with augmentation, 4) undoing the transformation based on a given sensor value 5) Subspace-Configurable Networks (SCNs) and 6) higher dimensional SCNs. More concretely, our contributions are:

- We show that data augmentation is not always the best strategy for fortifying ML models against data transformations, in particular with regard to resource constraints.
- We study alternative approaches to utilize auxiliary information, such as sensor data, to improve the accuracy of a model in the face of transformation of input data by up to 7.57% and achieve comparable accuracy with up 40.96% less parameters.
- We investigate how well the above-mentioned approaches work on a smartphone with constrained resources and show that for data adaptation the added overhead is only 0.03% of the total run-time.
- We show how well the different methods work with noisy sensor guidance.

We made our source code publicly available under https://github.com/CPUFronz/sensor_guided_adaptation.

2 Sensor-Guided Adaptation

Deep learning models are generally not robust to common input transformations, such as scaling or rotation of the input images. To counter this, deep learning models are usually trained with data augmentations. Data augmentation means that images in the training set get randomly perturbed during training. Fig. 2 illustrated how this could look like, by showing a sample image from the German Traffic Sign Recognition Benchmark (GTSRB) [20] dataset for different scale factors. The main drawback of this method is, that the neural networks storing these transformations require more parameters, meaning larger models and thus longer training and inference times.

Fig. 3 shows the relation between the parameter count and the accuracy on a randomly rotated CIFAR10 dataset [37] for different model types. Baseline, Inverse and the two DA approaches are all ShallowCNNs [40] consisting of 3 convolutional layers with 64 neurons followed by a fully connected layer with again 64 neurons. DA 2x and DA 4x instead have 128 and 256 neurons, respectively. The general trend in this plot suggests, that the bigger the model, the higher the accuracy. While this is acceptable in a cloud setting, smaller models are preferable in resource-constrained environments, such as the edge.

There exist different approaches to address the issue of making models robust against transformations of the input data, while not increasing the resource requirements. In the following paragraphs, we look at two approaches that utilize auxiliary sensor information. One approach is adapting the input data, while the other one adapts the model. Auxiliary information is readily available nowadays, *e.g.*, IMU sensor data from smartphones for rotation or distance derived from stereo vision for dual camera smartphones.

The first approach we study is to canonicalize the input image, by inverting the transformation for a given sensor value. For example, if a camera is mounted at a 10° angle, to work correctly, it has to be calibrated so that the image gets rotated by -10° before it is fed into the model, given that the model is trained on canonical images only. It is a simple approach, which can be applied with little modifications to the data pipeline. Compared to other approaches, we don't have to modify the model in this approach, only the data ingestion pipeline is slightly modified. The main advantages of this approach are, that the model does not have to be modified and implementing pre-processing functionality in an existing data pipeline is relatively simple. The main drawbacks are that the transformation has to be known in advance and invertible. Even if a transformation is invertible, it does not mean that it is fully reversible, *e.g.*, able to restore pixels that got lost during the transformation. Using the previous example again, when rotating an image, some parts at the edges are cut off and lost when the rotation is inverted.

The other approach utilizing auxiliary sensor information are Subspace-Configurable Networks (SCN) [29]. SCNs are a novel approach for resource-efficient adaptation of neural networks, utilizing sensor information to yield a network configuration that is optimized for a given input value. In our case, this input value is the value reported from a sensor. Each SCN consists of two major

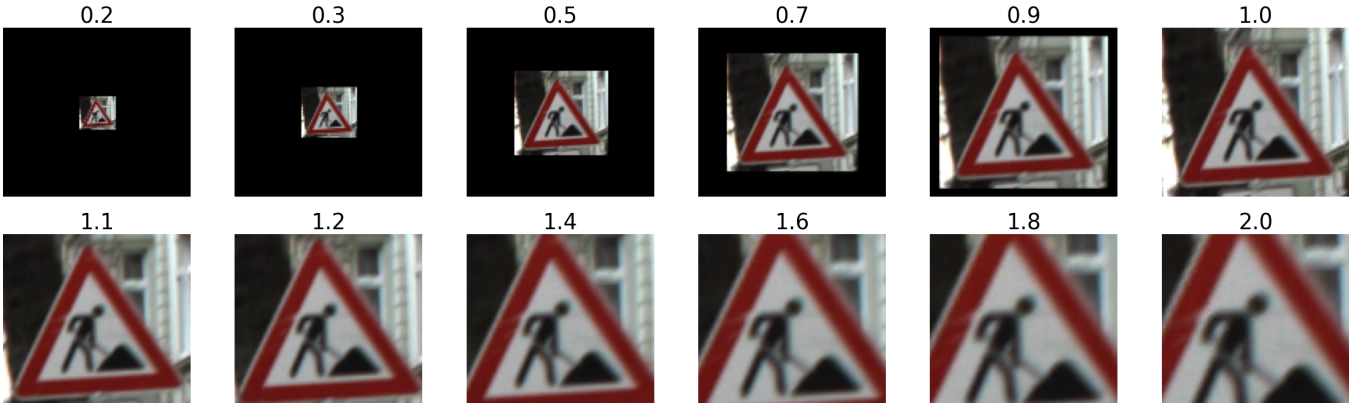


Figure 2: Sample image from the German Traffic Sign Recognition Benchmark (GTSRB) dataset [20] for different scale factors. For small-scale factors, many finer details get lost, while for scale factors larger than 1, the outer parts of the image are cropped therefore, the information contained in those parts is lost.

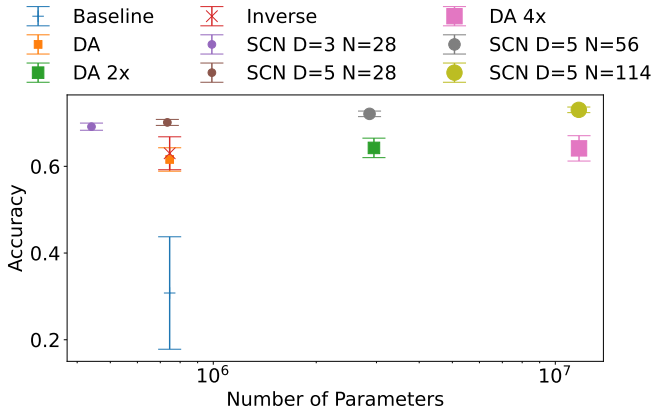


Figure 3: Number of parameters of a network and its accuracy for CIFAR10 Rotation for different approaches. The value D for SCN denotes the number of dimensions for the subspace, the value N denotes the width of the resulting inference network. We chose values for D and N so, that they are comparable to the number of parameters for DA.

components: a configuration network[2] and a D-dimensional configuration space from which the configuration network selects a weight configuration optimized for the given input value. In our scenario, the sensor data is used as input for the configuration network in order to yield the optimal configuration for the current transformation. Because of their resource efficiency and relative robustness, SCNs work well in IoT settings and other resource constraint environments.

3 Experiments

Datasets. We show our approaches on two different datasets: the German Traffic Sign Recognition Benchmark (GTSRB) dataset [20] consisting of 43 classes of traffic signs and the CIFAR10 dataset [37], consisting of 10 different classes. GTSRB consists of 51 839 images, split into 39 209 training images and 12 630 test images, and contains 43 classes of traffic signs. CIFAR10 consists of 60000 images of 10 classes, split into 50 000 training images and 10 000 test images. All images are resized to 32×32 pixels for our experiments, for CIFAR10, all images are 32×32 pixels by default, therefore we use them unchanged. For GTSRB, they have different image sizes. We scaled all images in GTSRB by a factor of 1.5 before further using them, significantly improving the performance, as those scaled images have unnecessary backgrounds removed and focus better on the traffic sign in the image.

Transformations. We use two transformations, which are among the most common transformations in real-world scenarios: rotation and scale. Rotation means an image is rotated around its center by an angle ϕ. In real-world scenarios, this could either occur statically when a camera mounting is fixed, similar to the example mentioned in section 1, or dynamically, *e.g.*, when a smartphone's camera is tilted. Scale means that the size of the image stays the same, but the content of the image is either scaled down (for scale factors $f < 1$) or up (for scale factors $f > 1$), similar to zooming in or out of an image. An example of scale is shown in Fig. 2. In our experiments, we train rotation with batch norm between the convolutional layers and scale without, because the different modes of batch norm (Train and Eval) can lead to train-inference mismatch [22], where the accuracy significantly drops during the evaluation of a model. Scale is more susceptible to train-inference mismatch than rotation, as even minor changes in scale can significantly alter the pixel distribution of an input image.[3]

Approaches. We compare six methods: baseline, inverse, training with data augmentation using models of different sizes, 3-dimensional and 5-dimensional Subspace-Configurable Networks, explained below.

(1) **Baseline:** The baseline model is trained on canonical images that are not transformed. With this model, we show how it would perform without any intervention. For both datasets,

[2]Also commonly referred to as Hypernetwork [9].

[3]Compare a scale factor of 0.2 and 1.0 in Fig. 2, for a scale factor 0.2 the distribution of pixel values is significantly different, because of the additionally added black pixels.

we use a 3-layer ShallowCNN [40] with 64 neurons in each convolutional layer [41], targeting the class of small models typically deployed on the resource-constrained edge.

(2) **Data augmentation (DA):** This approach has the same neural network architecture as Baseline, a ShallowCNN with 64 neurons, but is trained with augmentations. Each batch is scaled or rotated during training by a value drawn from a uniform random distribution.

(3) **Data augmentation double wide (DA 2x):** Like DA, this approach again uses augmentation, but with a wider network. We are again using a 3-layer ShallowCNN, we doubled the width, so this network has 128 neurons.

(4) **Inverse:** In this method, we undo the transformation by inverting it and using the resulting image as input for the Baseline model. For rotation, this means that if an image is rotated by an angle ϕ, we undo the rotation by rotating it by $-\phi$. For scale, we undo the transformation of a scale factor f, by scaling the transformed image with a scale factor of 1/f.

(5) **Subspace-Configurable Networks with a 3-dimensional subspace (SCN D=3):** As inference network for SCN we use again a 3-layer ShallowCNN, but this time with N=28 neurons. We choose this configuration, because it has a comparable accuracy to DA, with significantly fewer parameters (*cf.* Fig. 3). The configuration network for the SCN is a simple 2-layer neural network with 64 neurons and Softmax as the activation function for the output layer.

(6) **Subspace-Configurable Networks with a 5-dimensional subspace (SCN D=5):** Same as SCN D=3, but the subspace for the configuration network is 5-diemnsional, and the total number of parameters for this network is comparable to Baseline and DA.

We ran all our experiments for 5 different seeds and report standard deviations in the figures. For scale, we use a minimal scale value of $f = 0.2$ and a maximal value of $f = 2$, as this range covers all values, for which there is still significant information left in the resulting image. We work with 32×32 images, so $f = 0.2$ results in an image with 6×6 pixels, which is already a challenging task for all methods. Scaling up an image larger than $f = 2$, would also result in too much information being lost as only the central part of the image remains within the image boundary. For rotation, we use all 360°.

Noisy Sensors. In real-life deployments, sensor readings often tend to be noisy. Therefore, we investigate how this noisy data affects the accuracy of various approaches, aiming to understand its impact.

Runtime. Finally, we measure the latency for the different approaches on a smartphone. The smartphone used for the benchmarks was a Xiaomi Redmi Note 9 Pro with a Qualcomm Snapdragon 720G SoC and 6 GB of RAM. For benchmarking our models, we use the benchmark tool for TensorFlow Lite[4]. The models were originally trained using PyTorch, so we have to convert them to TensorFlow Lite. Inverse and SCN require additional steps during inference. In order to get a correct timing of each required step, we create faux-models, which perform the required steps, but are not making predictions as an actual model would. For Inverse, we create a faux-model to invert the transformation, for SCN we create two faux-models, one for the inference of the configuration Hypernetwork and one for the configuration of the model parameters, using the result of the Hypernetwork.

4 Results

Fig. 4 shows the accuracy of different approaches for rotation and scaling transformations for different transformation values. Fig. 4a shows a polar plot for the GTSRB data. Baseline works best for rotations close to 0° (*i.e.*, close to the distribution of the original dataset), and fails for other rotation angles. Inverse achieves the best average accuracy for all methods for Rotation on GTSRB with an average accuracy of 89.78%. DA and DA 2x achieve an average accuracy of 83.57% and 83.83% respectively. Compared to the accuracy of Baseline for 0° and Inverse, the accuracies for DA and DA 2x are lower, showing that data augmentation negatively affects the top accuracy of a model given a fixed model size. Both SCN models have a better accuracy compared to the two DA models, with a mean accuracy over all angles of 87.91% for D=3 and 88.3% for D=5.

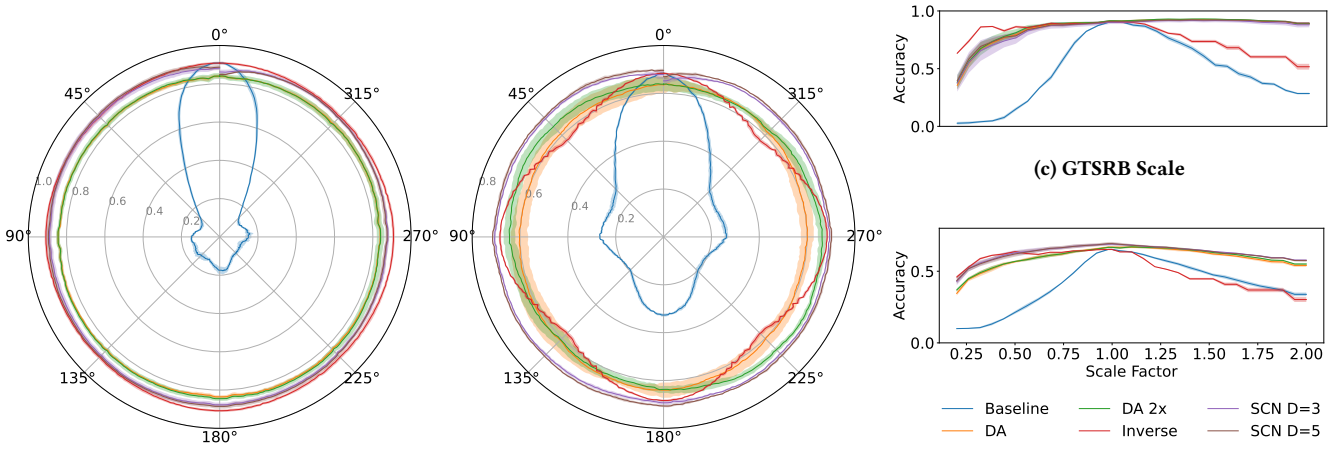
CIFAR10 is a more challenging dataset, as even the best of our methods only reach an average accuracy of around 70% with the ShallowCNN architecture, as shown in Fig. 4b. Baseline again only works for rotations around 0°, although its accuracy notably increases for multiples of 90°. Inverse conversely experiences notable drops in accuracy for multiples of 45°, due to pixels being cut off and irreversibly lost when applying and reverting the transformation. This results in an almost 10% difference in accuracy for rotations which are multiple of 45° compared to those which are multiples of 90 degrees[5]. Which, leads to an average accuracy of 63.01%, which is still better than DA with 61.58%, but slightly below the average accuracy of DA 2x with 64.25%. As the shading around the lines for both DA models indicate, there is some variability in the accuracy for both models. For each different seed, the resulting model converges toward a specific angle, which it works best for, at the expense of having less accuracy on an opposing angle. Both SCN models yield the best results for CIFAR10 rotation: D=3 has an average accuracy of 69.14% and D=5 of 70.12% and are thus better than Baseline without any rotation, which has an accuracy of 68.38%.

Fig. 4c shows how well the different methods perform on the GTSRB dataset for different scale factors. Baseline works again only for the base case. Inverse works well for a scale factor of 1, as well as for scale factors $f < 1$, for $f > 1$ the accuracy gradually decreases. This is due to when images with $f < 1$ get scaled up, the empty parts of the image get replaced by more meaningful pixels, conversely for $f > 1$ the previously cut-off pixels can not be restored and get replaced by black ones, see Fig. 2 for a visual representation. Due to this effect, Inverse only reaches an average accuracy of 77.91%, while DA and DA 2x reach an average accuracy of 85.71% and 86.25%. Both SCN models have a slightly lower average accuracy, with 84.67% for D=3 and 85.75% for D=5.

Scale for CIFAR10 follows a similar trajectory as for GTSRB, as shown in Fig. 4d, although Inverse for $f > 1$ has an even worse accuracy and even falls under the accuracy of Baseline for the same scale factor. This indicates, that the cut-off pixels are more important

[4]https://www.tensorflow.org/lite/performance/measurement, last accessed 2024-06-18

[5]The same is observed for GTSRB, but there the difference is only ~2%.



(a) GTSRB Rotation

(b) CIFAR10 Rotation

(c) GTSRB Scale

(d) CIFAR10 Scale

Figure 4: Accuracy of different methods for rotation and scaling for the GTSRB and CIFAR10 datasets. For both datasets and transformations for all angles and different scale factors, SCN approaches give the best accuracy, even in the case of D=3, which has 40.96% less parameters than DA.

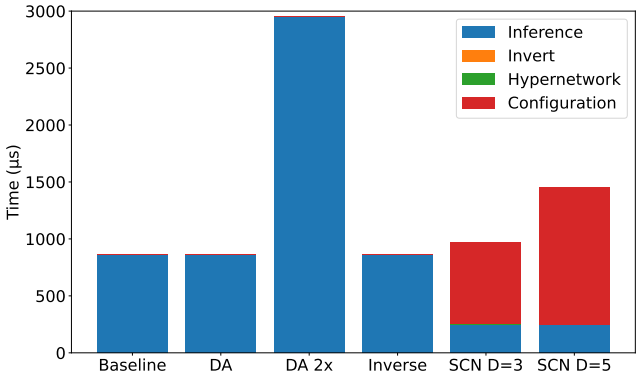
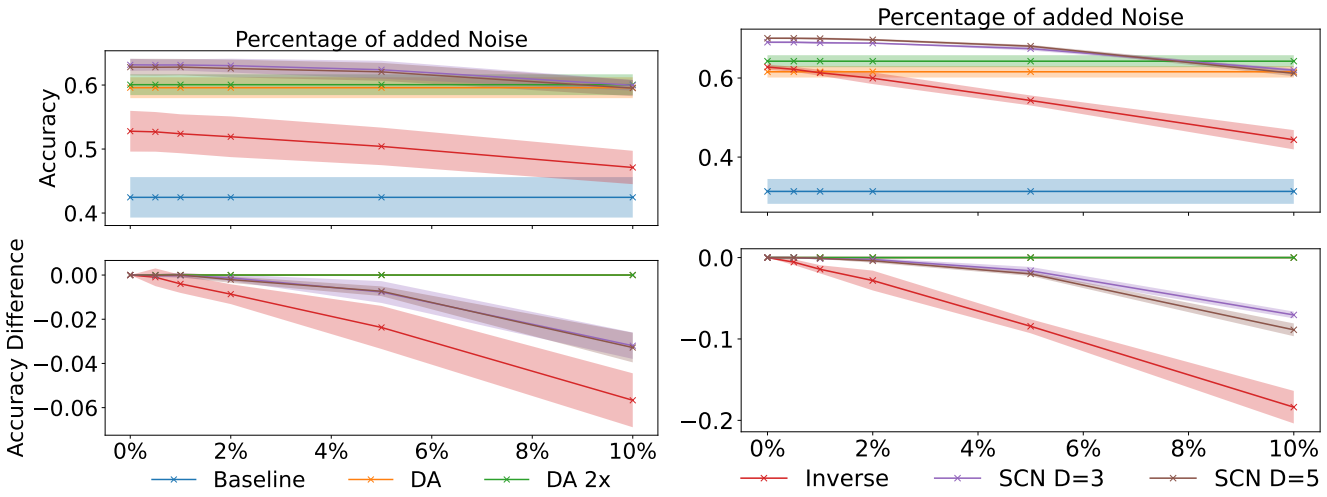


Figure 5: Average runtimes for the models. The overhead for inverting an image in Inverse is less than 1 μs and thus negligible compared to the inference time of the model. Both SCN models have a lower inference time, because their inference models are smaller than Baseline and DA. However, in the case of SCN D=5, the total parameter count is roughly the same as those two models, for SCN D=3 the parameter count is even lower, but the performance is still better than Baseline and DA, as shown in Fig. 4.

in CIFAR10, compared to GTSRB. DA and DA 2x have an average accuracy of 59.57% and 60.05% and are slightly below the average accuracy for both SCN models of 62.96% for D=3 and 62.84% for D=5. It is remarkable, that the smaller SCN model yields a slightly better performance, this indicates that the inference models of SCN are at capacity for the dataset.

Fig. 5 shows the average runtime for the different models on a Xiaomi Redmi Note 9 Pro phone with a Qualcomm Snapdragon 720G SoC and 6 GB of RAM. We are measuring rotation on CIFAR10 data, as it is the most challenging task. We measure the runtime of the models using the TensorFlow Lite benchmarking tool, and reporting the average runtime for all 5 seeds per model type. For models where the inference consists of more than one step, we measure and report each step individually. Baseline, Inverse and DA have the same model architecture and thus all have the same runtime of 861.59μs, 862.11 μs and 862.66 μs. Note, that we also measure the time to invert, but it can not be seen in the plot, as it only takes 0.26 μs. As the largest model, DA 2x takes 3.41 times longer than those models and has an execution time of 2950.08 μs. For both SCN models, we measure the inference time it takes the Hypernetwork to get the configuration parameters, the time it takes for the configuration to take place and the time for the inference of the resulting model. The time for the Hypernetwork inference is almost negligible, taking 1.59 μs for D=3 and 1.64 μs for D=5. Configuration time on the other hand causes significant overhead, taking 719.12 μs for D=3 and 1208.79 μs for D=5, which is much more time than the inferences, which take 250.34 μs and 245.88 μs respectively. The resulting inference networks only have 146 786 parameters and are thus 5.07 times smaller than the models used in Baseline, Inverse and DA, therefore they have faster inference time.

Fig. 6 shows the absolute accuracy and accuracy changes at different noise levels. The change is calculated by subtracting accuracy at each noise level from the 0% noise accuracy for the same transformations. Noise is drawn from a normal distribution to simulate real-world sensor noise, where the image is transformed by a given transformation value, while the value that is used in the models is this transformation value plus the noise. As expected, DA and DA 2x show no accuracy drop with increasing noise, while Baseline, though unaffected by noise, has lower accuracy with transformations. Inverse is most affected by sensor noise, with a 18.37% accuracy drop for 10% noise in rotation. SCN models also decline at 10% noise, but only by 7.04% for D=3 and 8.87% for D=5.



(a) CIFAR10 Scale **(b) CIFAR10 Rotation**

Figure 6: Accuracy and change in accuracy for different methods, when sensor data is noisy, for different percentages of added noise for CIFAR10. The accuracies for 0%, 0.5%, 1%, 2% 5% and 10% added noise, compared to sensor data without noise.

	Parameter Count	Model Size (MB)
Baseline	745 610	3
DA	745 610	3
DA 2x	2 949 386	11.8
Inverse	745 610	3
SCN D=3	146 786† / 440 513‡	0.59 † / 1.8 ‡
SCN D=5	146 786† / 734 103‡ /	0.59 † / 3 ‡

† – inference model ‡ – full model

Table 1: Number of parameters and corresponding sizes for different types of models for CIFAR10 rotation.

Scale transformations are inherently more robust against noisy sensors, and thus the decrease in accuracy is not as pronounced as for rotation. The accuracy of Inverse decreases by 5.66% for 10% of added sensor noise, while the accuracy for the two SCN models decreases by 3.19% and 3.28% respectively.

Table 1 shows the number of parameters and model size in MB for the different approaches. Note that for SCNs we report the numbers for the inference model and for the full model. The inference model is used for performing the actual inference, while the full model is the number of all parameters of the SCN, including the configuration network and the D-dimensional configuration space.

5 Discussion

As our results show, using data augmentation is not the only successful mitigation against input data transformation. Utilizing auxiliary information, such as sensor data, can significantly improve the accuracy of a model while requiring significantly less memory and computational resources. Canonicalization methods, such as inverting an input transformation, are simple to implement into existing data pipelines and add little to no performance overhead. Other approaches, such as SCN, add some overhead, but resulting models are able to achieve higher accuracy with fewer parameters, resulting in an execution time, only slightly higher than the base model. If the reconfiguration of the model is not necessary before inference, *e.g.*, when the rotation angle is static and does not change, so the SCN only has to do the configuration once, the models yielded from the configuration network of SCN would have the fastest inference times out of all the model types we studied in this work, as shown in Fig. 5.

Deploying larger models for data augmentation is not ideal for edge applications. However, leveraging readily available sensor data at the edge can greatly enhance the robustness of models against transformations. Inverting an input transformation has the disadvantage that the transformation has to be known in advance and that it can be inverted. DA and SCN are able to learn the transformation from data, although SCN requires a sensor value as input to guide the configuration network.

Sensor-guided methods are vulnerable to noisy sensor data, which can negatively impact their accuracy. The extent of this impact, however, depends on the type of transformation involved. Inverse is the most sensitive, while SCNs are more robust. DA, on the other hand, are robust towards noisy sensor data, as they don't rely on sensor guidance.

6 Related Work

Sensor Guided/Enhancing Machine Learning. SMiLe [34] is an automatic end-to-end framework to co-design and optimize sensing and ML for embedded systems, unlike our work the authors utilize sensing information during training via an AutoML approach. Poggi et al. [26] introduced a framework for resource-constrained embedded systems that enhances optical flow network accuracy by incorporating sparse, accurate hints from external sources. This framework, which modulates correlation scores using

depth measurements and hand-crafted algorithms, shows improved performance on standard benchmarks in both simulated and real-world conditions. Qi et al. [36] propose a multi-sensor guided hand gesture recognition system for surgical robot teleoperation using an LSTM model, which outperforms traditional methods in accuracy, speed, and robustness against interference in real-time tasks.

Model Adaptation. On-device model adaptation is a strategy to enhance deep learning model accuracy. In contrast to our approach, which does not involve on-device model re-training, Profentzas *et al.* [7] introduce MiniLearn, a framework enabling on-device re-training of pre-trained, quantized neural networks on resource-constrained IoT devices using locally collected data. MiniLearn notably accelerates inference and reduces memory usage compared to original DNNs. Another prominent demonstration of model adaptation through semantic segmentation is done by Sakaridis *et al.* [8]. The focus lies on enhancing nighttime image analysis without direct nighttime annotations. They utilize synthetic and real images to adapt daytime models, employing a curriculum framework that progressively transitions from day to night. [10] tackle model adaptation to variable resources by introducing resource-efficient deep subnetworks which make use of permutation invariance of neurons [30] to achieve efficient inference. Task Vectors [13], are a novel approach for editing models, where pre-trained models are adapted by adding or subtracting difference-vectors from a pre-trained and a fine-tuned model.

Data Augmentation. Data Augmentation is often presented as a solution to the challenge of limited data, particularly for domains where big data is not readily available, such as medical image analysis. In [43], diverse augmentation techniques, such as geometric transformations, color space augmentations, kernel filters, and adversarial training with a focus on Generative Adversarial Networks (GANs) are explored. The survey compares aspects like test-time augmentation, resolution impact, final dataset size, and curriculum learning, providing insights into enhancing model performance and maximizing the potential of limited datasets. Training with data augmentation is one approach to make models robust, insensitive, or invariant to specific transformations of the input data [4, 18]. The work by He *et al.* [21] utilizes an approach, where a random area of an image is magnified using a lightweight, model-free data augmentation method called Local Magnification (LOMA). This method improves image classification and object detection performance on different datasets. Other data augmentation techniques include enhancing conventional input augmentation by applying it at the feature level, as presented in [5]. A convolutional layer is randomly selected for each mini-batch and applies independent affine transformations to each activation map, introducing a novel approach to improve neural network stability. Mounsaveng *et al.* [31] present a fully differentiable, end-to-end data augmentation network that improves image classifier performance by learning general transformations through an adversarially trained Spatial Transformer Network within an encoder-decoder architecture, showing that this approach is more effective than generating images from scratch or separating augmentation and classification tasks.

Data Canonicalization. Kaba *et al.* [33] introduce symmetry-based neural networks for achieving equivariance in neural networks by learning canonicalization functions, which map inputs to canonical representations. This approach provides universality and interpretable insights, suggesting that training small neural networks for canonicalization outperforms predefined heuristics. Schmidt and Stober [42] address vision model sensitivity to image rotations with a radial beam sampling strategy and radial kernels for rotation equivariance. Their model, radial beam-based image canonicalization (BIC), incorporates center-rotation covariance, enabling rotation-invariant vision pipelines and enhancing performance across various datasets.

Boosting Performance with Additional Information. There are different ways to use auxiliary information to boost the performance of deep learning models. Integrating feature attribution methods into deep network training, known as attribution priors, can optimize models for desirable properties, such as the importance or unimportance of particular features. In [12], attribution priors focused on higher-level properties like smoothness and sparsity, utilizing a new fast attribution method called expected gradients that satisfies fundamental interpretability axioms, are utilized. The approach improves model performance across diverse real-world tasks. The widespread success of DNNs in various domains has challenged deploying these accurate models for practical ML solutions [27]. The computational expense, power consumption, and large memory requirements of deep learning algorithms often lead to cloud-based processing, resulting in high latency and privacy concerns. To address this, the article explores four research directions for efficient deep learning inference on edge devices, including novel deep learning architecture, optimization methods, algorithm-hardware co-design, and efficient accelerator design, aiming to review tools and techniques for enhancing edge inference.

Resource Constrained Devices. Resource-constrained IoT devices encounter performance challenges like latency, communication costs, and privacy issues associated with data offloading to external servers [23]. Efforts have focused on deploying ML systems on edge computing devices proximal to IoT devices, aiming to address these challenges. The survey explores various research endeavors in this domain, highlighting operational aspects such as compression techniques, tools, frameworks, and hardware in successful applications of intelligent edge systems. In addition to deploying ML systems on edge devices, alternative training methods aim to enhance neural network performance. Cai *et al.* [14] propose Network Augmentation (NetAug), a novel training method tailored for small neural networks. Unlike traditional regularization techniques, NetAug combats underfitting by augmenting networks through reverse dropout, embedding tiny models within larger ones to gain additional supervision. This method enhances tiny model performance for image classification and object detection, achieving up to a 2.2% accuracy boost on ImageNet and requiring significantly fewer Multiply-Accumulate operations for object detection on Pascal VOC and COCO datasets. Prior to NetAug, Dai *et al.* [17] investigated deformable convolution and deformable RoI (region-of-interest) pooling to improve CNNs' geometric transformation modeling. These modules learn spatial sampling offsets, integrate easily into existing CNNs and are trainable end-to-end. Experiments showed their effectiveness in tasks like object detection and semantic segmentation. Model-Adaptive Data Augmentation (MADAug) [6] is a more recent approach, which trains an augmentation policy network to adapt strategies for each input image and

training stage. MADAug outperforms existing methods in various image classification tasks and architectures. Initially, the model is trained without augmentation, and data augmentation is gradually introduced.

Invariance. Invariance in data augmentation is crucial for consistent predictions despite input variations. Atienza *et al.* [1] introduce Agreed Maximization (AgMax), a simple regularization method that improves generalization by aligning predictions from two transformed images of the same label. Empirical results show significant performance gains across classification, detection, and segmentation, outperforming other augmentation methods. Jaderberg *et al.* [24] introduced the Spatial Transformer module, a learnable component that enables neural networks to perform spatial transformations on feature maps without extra supervision or changes to optimization. Integrating this module into CNNs improves performance by imparting invariance to translation, scale, rotation, and other deformations. The transformation parameters can be extracted and applied in later tasks, highlighting its versatility in both feed-forward and recurrent networks. Benton et al. [11] investigate discovering invariances and equivariances from training data with Augerino, which enhances these properties in neural networks via data augmentation. Augerino learns a distribution over augmentations, optimizing both network and augmentation parameters, and supports various symmetry groups. It integrates seamlessly with standard architectures and optimization algorithms, showing superior performance in regression, classification, and segmentation tasks across image and molecular data. An extension by Mahan et al. [32] introduces class-specific augmentation distributions, allowing models to adapt to different symmetries within complex datasets, improving generalization across diverse tasks.

7 Conclusion

In this work, we study how different approaches can mitigate transformations and perturbations of input data for deep learning models. We compared how data augmentation performs compared to sensor-guided approaches. Sensor-guided approaches can outperform data augmentation while requiring fewer resources, making them a good choice for resource-constrained devices.

Acknowledgments

This work has been partly supported by the FFG-COMET-K1 Center Pro²Future (Products and Production Systems of the Future), Contract No. 881844 (CORVETTE project). The authors would also like to thank Dong Wang for his insights and help regarding SCNs. The results presented in this paper were computed using computational resources of the Zentraler Informatikdienst of Graz University of Technology.

References

[1] R. Atienza. 2021. Improving Model Generalization by Agreement of Learned Representations from Data Augmentation.
[2] V. Biscione and J.S. Bowers. 2021. Convolutional Neural Networks Are Not Invariant to Translation, but They Can Learn to Be. *JMLR* 22, 229 (2021).
[3] N. Cao and O. Saukh. 2023. Geometric Data Augmentations to Mitigate Distribution Shifts in Pollen Classification from Microscopic Images. arXiv:2311.11029 [cs.CV] https://arxiv.org/abs/2311.11029
[4] A. Botev *et al.*. 2022. Regularising for Invariance to Data Augmentation Improves Supervised Learning.
[5] A. Sandru *et al.*. 2022. Feature-Level Augmentation to Improve Robustness of Deep Neural Networks to Affine Transformations.
[6] C. Hou *et al.*. 2023. When to Learn What: Model-Adaptive Data Augmentation Curriculum. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
[7] C. Profentzas *et al.*. 2023. MiniLearn: On-Device Learning for Low-Power IoT Devices. In *Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks (EWSN '22)*.
[8] C. Sakaridis *et al.*. 2019. Guided Curriculum Model Adaptation and Uncertainty-Aware Evaluation for Semantic Nighttime Image Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
[9] D. Ha *et al.*. 2016. HyperNetworks.
[10] F. Corti *et al.*. 2024. REDS: Resource-Efficient Deep Subnetworks for Dynamic Resource Constraints.
[11] G. Benton *et al.*. 2020. Learning Invariances in Neural Networks from Training Data. In *Advances in Neural Information Processing Systems*, Vol. 33.
[12] G. Erion *et al.*. 2021. Improving Performance of Deep Learning Models with Axiomatic Attribution Priors and Expected Gradients. *Nature Machine Intelligence* 3, 7 (July 2021).
[13] G. Ilharco *et al.*. 2023. Editing Models with Task Arithmetic.
[14] H. Cai *et al.*. 2022. Network Augmentation for Tiny Deep Learning.
[15] H. Chu *et al.*. 2024. nnPerf: Demystifying DNN Runtime Inference Latency on Mobile Platforms. In *Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems (SenSys '23)*.
[16] H. Wu *et al.*. 2021. CvT: Introducing Convolutions to Vision Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.
[17] J. Dai *et al.*. 2017. Deformable Convolutional Networks. In *Proceedings of the IEEE International Conference on Computer Vision*.
[18] J. Geiping *et al.*. 2023. How Much Data Are Augmentations Worth? An Investigation into Scaling Laws, Invariance, and Implicit Regularization.
[19] J. Kaplan *et al.*. 2020. Scaling Laws for Neural Language Models.
[20] J. Stallkamp *et al.*. 2011. The German Traffic Sign Recognition Benchmark: A Multi-Class Classification Competition. In *The 2011 International Joint Conference on Neural Networks*.
[21] K. He *et al.*. 2022. Local Magnification for Data and Feature Augmentation.
[22] K. You *et al.*. 2024. Efficient ConvBN Blocks for Transfer Learning and Beyond.
[23] M. G. Murshed *et al.*. 2021. Machine Learning at the Network Edge: A Survey. *Comput. Surveys* 54, 8 (Oct. 2021).
[24] M. Jaderberg *et al.*. 2015. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems*, Vol. 28.
[25] M. Maszkiewicz *et al.*. 2017. Fine Guidance Sensor/near-Infrared Imager and Slitless Spectrograph on James Webb Space Telescope: Pupil Alignment Methodology and Metrology. In *International Conference on Space Optics — ICSO 2016*, Vol. 10562.
[26] M. Poggi *et al.*. 2021. Sensor-Guided Optical Flow. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*.
[27] Md.M.H. Shuvo *et al.*. 2023. Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review. *Proc. IEEE* 111, 1 (Jan. 2023).
[28] M. Yu *et al.*. 2023. Revisiting Data Augmentation in Model Compression: An Empirical and Comprehensive Study.
[29] O. Saukh *et al.*. 2023. Subspace-Configurable Networks.
[30] R. Entezari *et al.*. 2022. The Role of Permutation Invariance in Linear Mode Connectivity of Neural Networks.
[31] S. Mounsaveng *et al.*. 2019. Adversarial Learning of General Transformations for Data Augmentation.
[32] S. Mahan *et al.*. 2021. Rotating Spiders and Reflecting Dogs: A Class Conditional Approach to Learning Data Augmentation Distributions.
[33] S.-O. Kaba *et al.*. 2023. Equivariance with Learned Canonicalization Functions. In *Proceedings of the 40th International Conference on Machine Learning*.
[34] T. Goyal *et al.*. 2023. SMiLe: Automated End-to-end Sensing and Machine Learning Co-Design. In *EWSN (EWSN '22)*.
[35] T. Hahn *et al.*. 2019. Self-Routing Capsule Networks. In *Advances in Neural Information Processing Systems*, Vol. 32.
[36] W. Qi *et al.*. 2021. Multi-Sensor Guided Hand Gesture Recognition for a Teleoperated Robot Using a Recurrent Neural Network. *IEEE Robotics and Automation Letters* 6, 3 (July 2021).
[37] A. Krizhevsky and G. Hinton. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).
[38] A. Mumuni and F. Mumuni. 2021. CNN Architectures for Geometric Transformation-Invariant Feature Representation in Computer Vision: A Review. *SN Computer Science* 2, 5 (June 2021).
[39] A. Mumuni and F. Mumuni. 2022. Data Augmentation: A Comprehensive Survey of Modern Approaches. *Array* 16 (Dec. 2022).
[40] B. Neyshabur. 2020. Towards Learning Convolutions from Scratch. In *Advances in Neural Information Processing Systems*, Vol. 33.
[41] K. O'Shea and R. Nash. 2015. An Introduction to Convolutional Neural Networks.
[42] J. Schmidt and S. Stober. 2023. Learning Continuous Rotation Canonicalization with Radial Beam Sampling.
[43] C. Shorten and T.M. Khoshgoftaar. 2019. A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6, 1 (Dec. 2019).