Sparse is the new dense: improved generalization in neural network classifiers without retraining

Ahmad Ayaz Amin

Parkdale Collegiate Institute

ayaza2004@outlook.com

Abstract

The neocortex is the component of the mammalian brain that deals with common sense, abstract reasoning, and utilises prior knowledge to perform actions. It is the 'intelligent' portion of the brain, that is, it performs actions based on reasoning rather than instincts. Modern deep neural networks are loosely based on the feedforward connections in the neocortex, but do not prescribe any reasoning to perform actions. The side effect of this is that neural networks are very easy to break, as in the case of adversarial attacks. In this paper, we show how a recent modification to neural networks can significantly increase the generalization capabilities of neural networks, especially against adversarial attacks on various datasets. We do this by training the new architecture on clean images and compare its performance on corrupt images against baseline neural networks. The results show that the sparse architecture outperforms the standard architectures on most experiments.

Introduction

Humans can intelligently perform actions based on reasoning. A child needs to see a picture of a cat only once to recognize it in different scenarios, whether it is hiding behind a bush, or even if it is a different breed. This ability to reason in different scenarios with confidence [1] stems from the most advanced portion of the mammalian brain: the neocortex. The past few years have shown the amazing accomplishments of contemporary deep learning. Algorithms based on deep neural networks have shown unprecedented results in fields ranging from image classification [2], to superhuman performance on Atari games [3] to beating world class players in the board game Go [4]. Nonetheless, modern deep neural network systems do not pose a lot of generalization. For instance, an agent that has trained for over one million frames on Atari Breakout fails to perform as well on minor perturbations, such as a higher paddle [5]. Additionally, deep reinforcement learning is not able to solve all tasks, as in the game of Montezuma's Revenge [3]. An extreme example of such are adversarial attacks [6], which changes the input data ever so slightly that it fools neural networks, despite the high-level

features remaining indistinguishingly the same to the human eye. This renders neural networks as unusable for many real-world purposes, including security and autonomous driving.

Convolutional neural networks (CNNs) are the backbone of recent image-based tasks, and they outperform simple fully connected neural networks in such tasks due to their use of kernels from image processing literature, which mimic the receptive field of the visual system. However, such networks also have major drawbacks. The pooling layer in CNNs are especially notorious for losing valuable information, causing these networks to ignore relational information [7]. Moreover, the dense representations learned by standard CNNs can store unnecessary information that can cause a decrease in accuracy on corrupt images [8].

In this work, we explore how a novel modification [8] creates robust neural networks that can withstand corrupt data of various types. In particular, the novel architecture is able to withstand adversarial examples better than dense networks by a large margin without retraining, as well as retain high accuracies on occluded test images.

Related Work

Many labs have taken a biological approach to developing intelligent machines that can generalize to different scenarios. The Recursive Cortical Network (RCN) [9] probabilistic graphical model can generalize from few training images, and the Hierarchical Temporal Memory (HTM) [10] is resistant to noise on various data types. Both models are inspired by the mammalian neocortex, which in contrast to deep neural networks, generalize better without retraining. However, the RCN requires clean data to train on and the HTM has yet to show any state-of-the-art results. Furthermore, the RCN stores prototypes of its training examples in memory, making it intractable for use with large ImageNet [11] sized datasets, and the HTM requires binary input data, making it hard to apply it to real valued image data. Deep neural networks do not face such problems. Recently, Numenta has performed some experiments with deep neural networks by adding spatial pooling to add sparsity to the network [8]. The results show that the modified network outperforms standard deep neural networks on different noise levels from the MNIST dataset by a large margin. It is hypothesized that the same architecture will be able to resist adversarial examples, as well as occluded images while maintaining high accuracies.

In this work, we extend the previous work [8] by investigating the robustness of sparse neural networks across harder scenarios. We investigate the performance of sparse neural networks by benchmarking against baselines, which are standard dense networks.

Despite sharing similar architectures, sparse neural networks [8] differ greatly from standard neural networks in terms of their properties. The most basic sparse neural network layer is a linear block that is made to store sparse information using spatial pooling, whereas standard layers contain densely stored information. Furthermore, sparse neural networks make use of k-winners-take-all mechanisms in the hidden layers of a network, while standard networks use traditional activation functions, such as rectified linear unit (ReLU). The benefit of using sparse neural networks is that the network only needs to learn the salient properties of the data in order to perform

its task, making it robust against noisy images [8]. This is also why sparse neural networks train faster than dense neural networks [8].

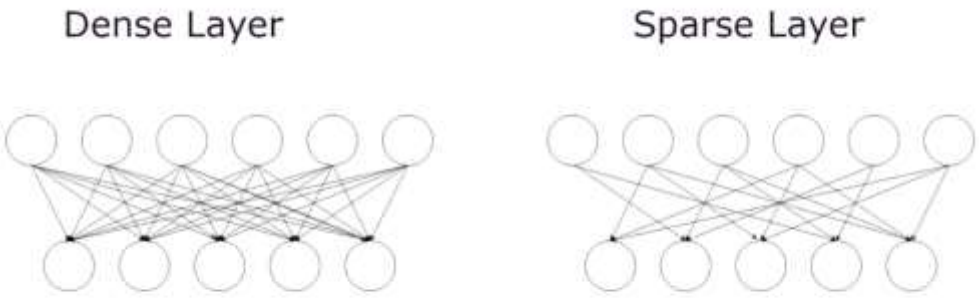

Figure 1 Left: Dense neuron. The layer takes all the information from the previous layer and feeds it to the next. Right: Sparse neuron. The layer receives only the most important information from the previous layer and sends messages to the next layer via spatial pooling.

Constructing Sparse Neural Networks

Implementing sparse neural networks is similar to constructing dense networks: a graph is defined and trained by backpropagating gradients. The main difference between sparse networks and dense networks is that the underlying operations are modified to induce sparsity. For a standard image classifier consisting of convolutions, linear operators and activation functions, the following modifications are to be made:

- All convolutions must be replaced with sparse convolutions.
- All hidden linear operators must be replaced with sparse linear operators.
- All activation functions succeeding hidden convolution and linear operators must be replaced with the k-winners-take-all function.
- The last linear operator and activation function must remain dense. This is to map features to labels in the final classification step.

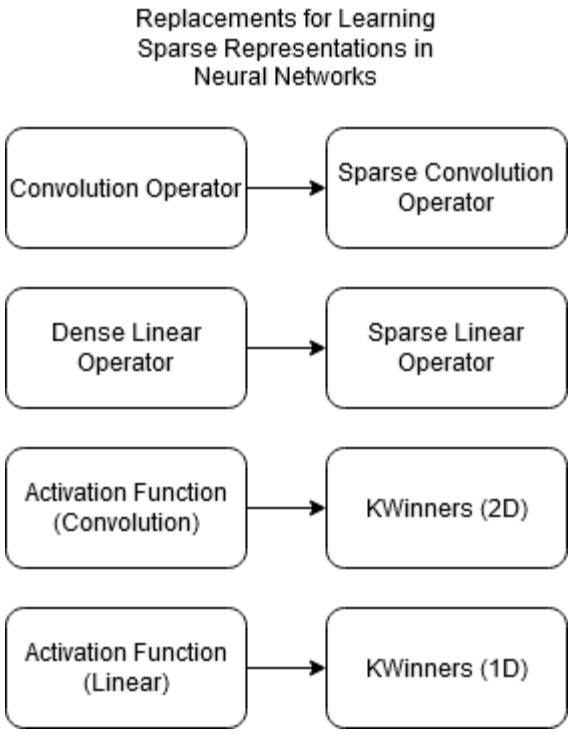

Figure 2 Module replacements required to induce sparsity.

Experimental

For the first experiment, we benchmark a Sparse CNN against a baseline CNN for classification. Both networks consist of two convolution layers, a hidden layer, and finally a linear layer with a softmax activation function, with the Sparse CNN containing the necessary modules for adding sparsity. We train the two models on the MNIST dataset, which we split into 50,000 training images and 10,000 testing images. The models are trained with a batch size of 32 for 10 iterations. We record the testing accuracy and loss for both networks, and then test their accuracies on corrupt variations of the test images. We use 10,000 images corrupted with noise, 10,000 images with boundary boxes, 1,000 images with occlusion and 10,000 images with adversarial noise. We use Fast Gradient Sign Method (FGSM) [12] and Projected Gradient Descent (PGD) [13]to corrupt the test images for adversarial examples.

For the second experiment, we benchmark classification performance of Sparse CNN against a baseline CNN on the Street View House Numbers (SVHN) dataset, a significantly harder problem for generalization. Both CNN architectures are SqueezeNets [14], with the Sparse CNN containing the necessary modifications to learn sparse features. The models were trained only on the train split of the SVHN dataset and tested on adversarial examples from the test set. The models were trained using a batch size of 32 for 20 iterations and testing was done with the FGSM adversarial attack.

Results and Discussion

For the first experiment, both networks attained state-of-the-art accuracy after training for 10 iterations. The Sparse CNN had a much lower test loss than the baseline Figure 3, whereas the latter had a better accuracy overall Figure 4.

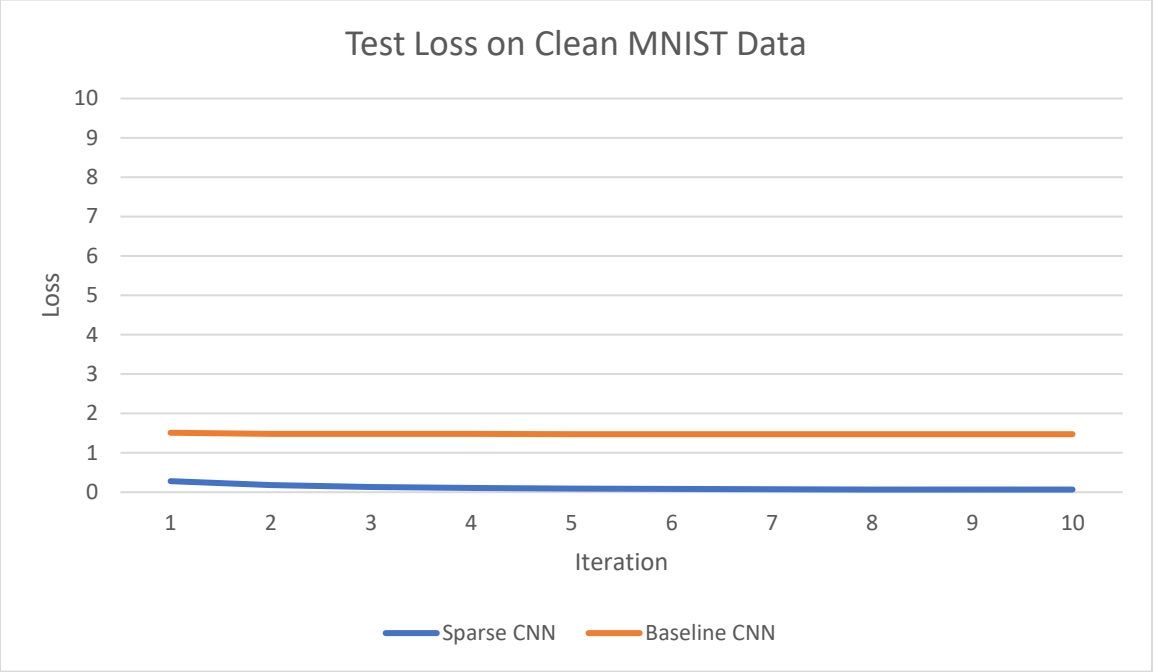

Figure 3 Test loss of Sparse CNN and baseline CNN on clean MNIST images.

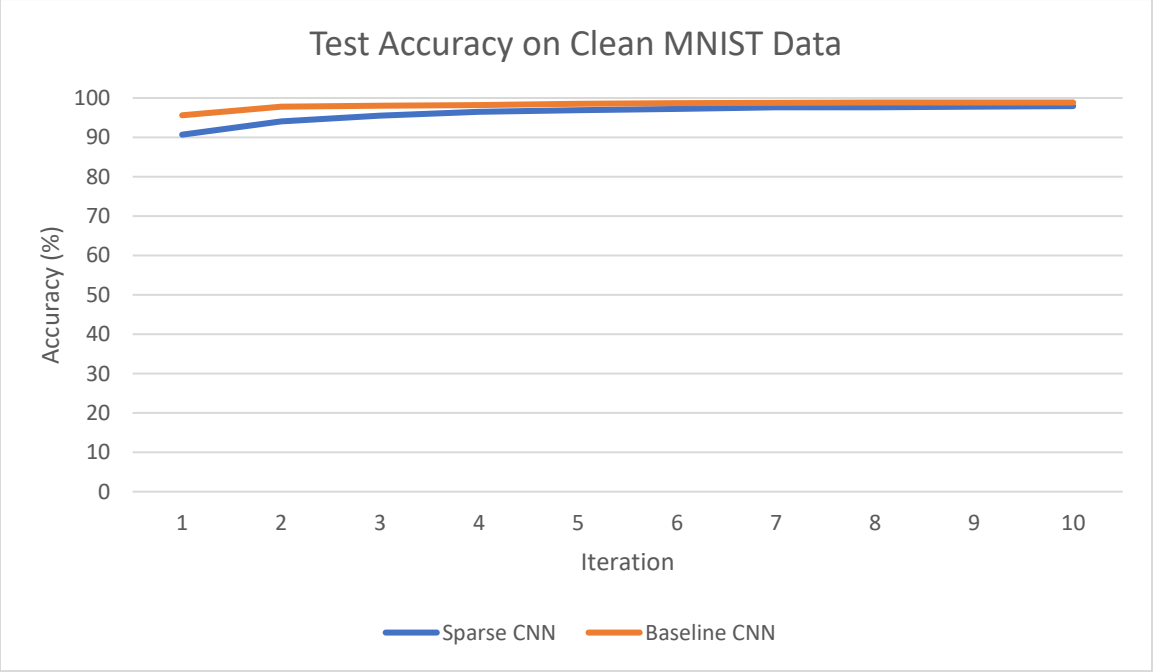

Figure 4 Test accuracy of Sparse CNN and baseline CNN on clean MNIST images.

The slow convergence to accuracy in the Sparse CNN was due to the spatial pooling, which caused the model to search for useful representations to base its classification. This is also the reason for the lower final loss in the Sparse network Figure 3. Both the Sparse CNN and the baseline CNN showed stable accuracies on noisy images since the noise factor was too low to show any changes in network stability. For occluded images, the Sparse CNN performed much better, which is due to the inductive bias formed from sparse coding. The baseline CNN had better performance with boundary boxes, which could be due to the resulting image being smaller, making it easier for the baseline network to classify successfully. Sparse representations are found to be especially resistant to adversarial attacks. The Sparse CNN is able to maintain > 76 % accuracy for FGSM examples and > 56 % accuracy for PGD examples, beating the baseline network by over a factor of 5 on FGSM and over a factor of 50 for PGD. This is due to the result of the sparse representations ignoring pixel level features, instead focusing on the higher-level aspects of the image Figure 5.

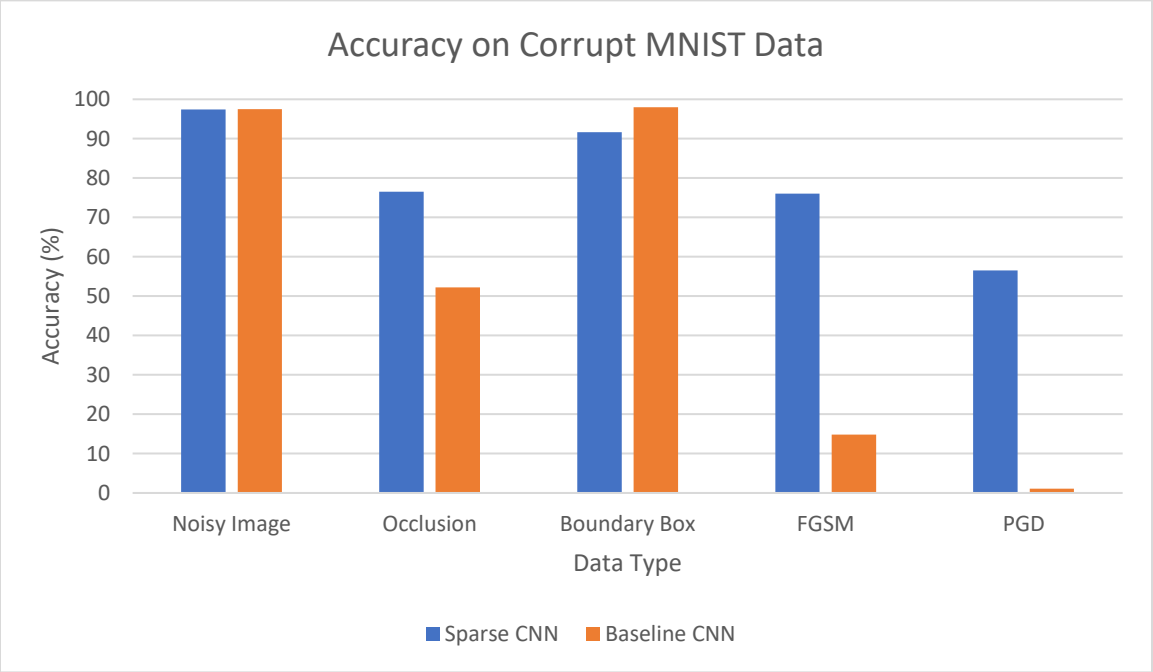

Figure 5 Test accuracies of Sparse CNN and baseline CNN on corrupt MNIST images without retraining.

For the second experiment, we benchmark a Sparse CNN against a baseline CNN on the SVHN dataset. Unlike the MNIST experiment, in which both the Sparse CNN and baseline CNN converge to state-of-the-art accuracies, the networks for the SVHN experiment achieve average training

losses of 2.312 and 1.4365 respectively Figure 6.

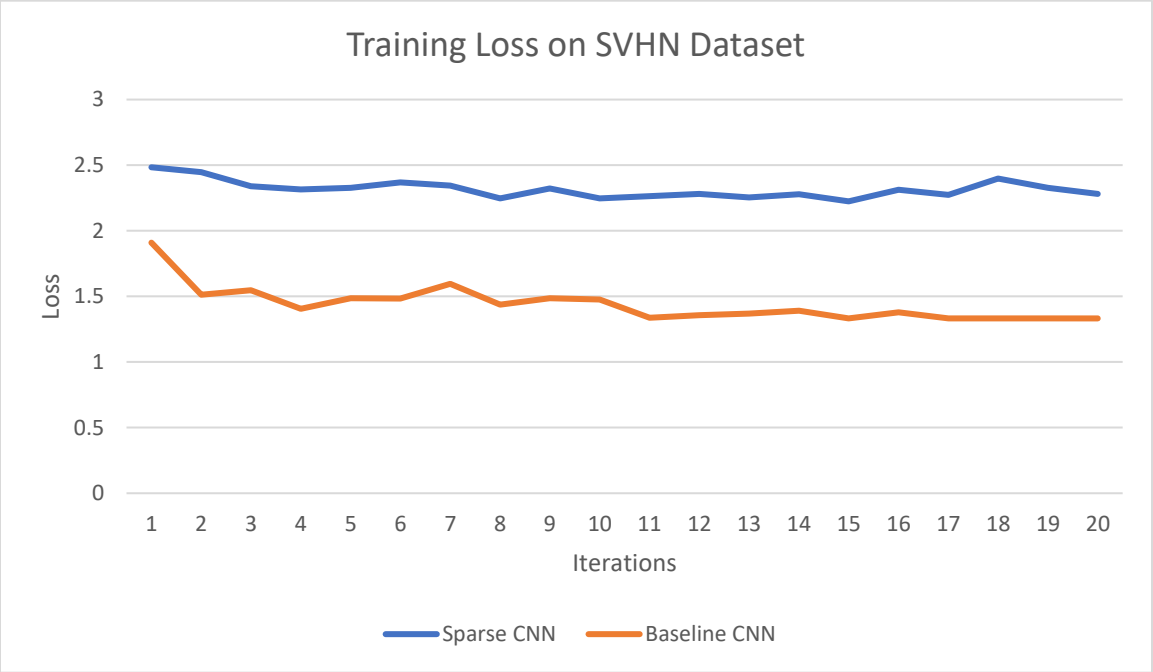

Figure 6 Training losses of Sparse CNN and baseline CNN on the SVHN dataset.

We did not expect the networks to achieve state-of-the-art accuracies since SVHN is a much harder problem to solve, requiring specialized architectures to achieve impressive results. However, the surprising result was that the SqueezeNets – both sparse and dense – were performing very poorly and were not able to converge. This could be due to faults in the SqueezeNet implementation (the baseline used was the official Pytorch implementation, with the Sparse model being modified to ensure that it matched the former as close as possible). Nonetheless, the test results for the adversarial attacks were also poor.

Ground Truth Label: 5

Epsilon Level

	0.05	0.1	0.15	0.2	0.25	0.3
Sparse CNN	1	1	1	1	1	1
Dense CNN	3	3	3	5	5	5

Ground Truth Label: 2

Epsilon Level

	0.05	0.1	0.15	0.2	0.25	0.3
Sparse CNN	1	1	1	1	1	1
Dense CNN	2	3	2	2	2	2

There were some interesting observations. Although both networks did not converge to an optimal solution, the dense network managed to classify the images correctly in most of the adversarial attacks. The Sparse CNN did not classify the instances correctly, but still kept the same prediction for all the adversarial attacks. For the first image, the baseline network began to classify the instance correctly after the image has increased in adversarial noise to epsilon 0.2, whereas for the second image, it misclassified the image at epsilon level 0.1.

For the third experiment, we trained another network on the EMNIST dataset. As the MNIST dataset is considered a solved benchmark, it makes sense to record performance against a more complex variant of the dataset. The EMNIST dataset contains letters alongside digits, forcing algorithms to consider more possible values for classifying novel images. The network is based of AlexNet [2], with the convolutional feature extractor modified with sparse convolutions and k-winner-take-all activations. The final classifier remains the same. The network is then trained on

the ‘balanced’ dataset for a total of 10 iterations with a batch size of 32 images.

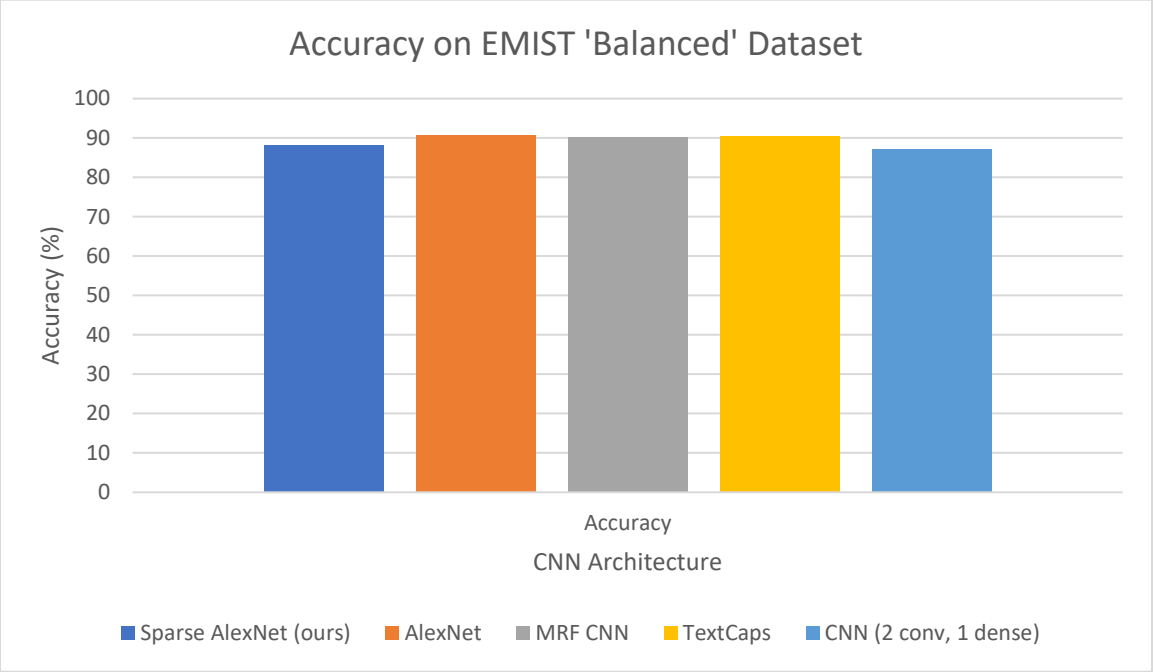

Figure 7 Accuracies of different convolutional networks on the 'balanced' set of the EMNIST dataset. The accuracies of the baseline networks were taken from [15].

The network achieved a final accuracy of 88.24 %, which is better than the CNN with 2 convolutions and 1 dense layer, but worse than the MRF CNN, TextCaps and (dense) AlexNet. We did not expect Sparse AlexNet to achieve higher accuracies than its dense counterpart, but we did not expect it to perform worse than most of the other baselines.

Overall, Sparse Networks were shown to perform nearly as well as standard networks, while achieving better generalization in scenarios where the data is corrupt.

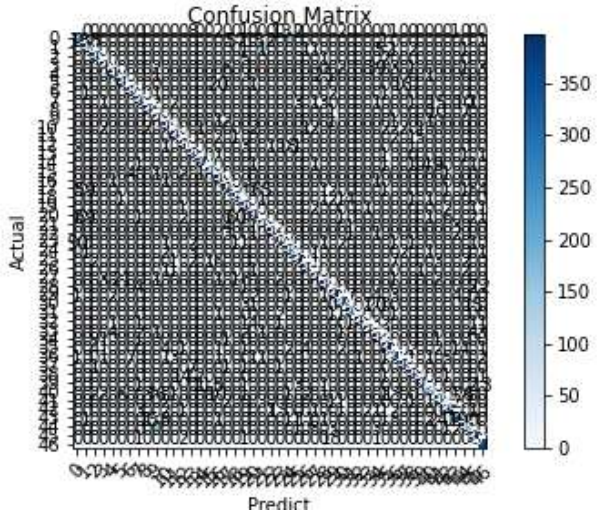

Figure 8 Confusion matrix of Sparse AlexNet on the 'balanced' set of the EMNIST dataset.

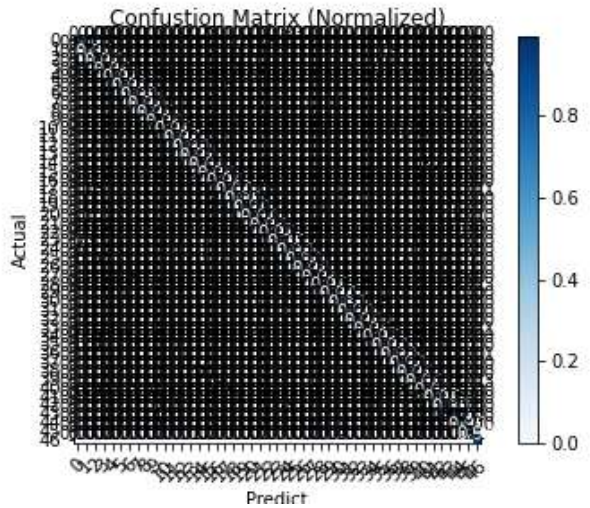

Figure 9 Normalized confusion matrix of Sparse AlexNet on the 'balanced' set of the EMNIST dataset.

Conclusion

Sparse representations help neural networks to generalize to corrupt data without retraining, especially against adversarial attacks. Sparsity allows useful features to be learnt in the network, which it uses to perform classification while maintaining high accuracies. This is especially important for use in the real world, as adversarial attacks against neural networks can cause major problems. For example, autonomous vehicles require a vision system that is able to generalize, since a faulty computer vision system may ultimately lead to accidents. A general vision system is also important for building general systems, as it is a major component for creating systems that can perform zero-shot task transfer [16].

Acknowledgements

We used PyTorch [17] to build, train and test all of the models, and the open-source PyTorch library NuPIC.torch [8] to evaluate the sparse models in the experiment. We used the PyCM [18] library to generate the confusion matrix for the EMNIST experiment. The occluded, noisy, and bounded MNIST datasets used in the experiment were downloaded from the RCN blogpost [19].

References

[1] M. Zwaan, "Embodied sentence comprehension," pp. 224-245, 2005.

[2] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," 2017.

[3] V. Minh, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature,* 2015.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Dreissche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature,* 2016.

[5] K. Kansky , T. Silver , D. A. Ḿely, M. Eldawy , M. Ĺazaro-Gredilla, X. Lou, N. Dorfman , S. Sidor , S. Phoenix and D. George, "Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics," 2017.

[6] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing properties of neural networks," 2014.

[7] S. Sabour, N. Frosst and G. E. Hinton, "Dynamic Routing Between Capsules," p. 2, 2017.

[8] S. Ahmad and L. Scheinkman, "How Can We Be So Dense? The Benefits of Using Highly Sparse Representations," 2019.

[9] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin and D. S. Phoenix, "A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs," *Science,* 2017.

[10] Y. Cui, S. Ahmad and J. Hawkins, "The HTM Spatial Pooler – A neocortical algorithm for online sparse distributed coding. Frontiers in Computational Neuroscience," 2017.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009.

[12] I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and Harnessing Adversarial Examples," 2014.

[13] A. Kurakin, I. Goodfellow and S. Bengio, "Adversarial Machine Learning at Scale," 2016.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016.

[15] A. Baldominos , Y. Saez and P. Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *MDPI Applied Sciences,* 2019.

[16] M. Lázaro-Gredilla, D. Lin, G. Swaroop and D. George, "Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs," *Science,* 2019.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[18] S. Haghighi , M. Jasemi , S. Hessabi and A. Zolanvari, "PyCM: Multiclass confusion matrix library in Python," *Journal of Open Source Software,* vol. 3, no. 25, p. 729, 2018.

[19] "Common Sense, Cortex, and CAPTCHA," 2017.