Comparison of Multilayer Perceptron and Convolutional Neural Network in hand-written digit recognition

Abstract

This paper evaluates multilayer perceptron (MLP) and a convolutional neural network (CNN) applied to hand-written digit recognition. Both algorithms were trained with varying parameters such as number of layers and training set sizes. As expected, CNN outperforms MLP, however once the training dataset increases in size, the difference in the accuracy between both algorithms is relatively small with MLP accuracy increasing by nearly 10% compared to under 3% for CNN. Interestingly, both performed surprisingly well even if trained under the presence of significant label noise.

1. Introduction

In this project we compare the performance of a MLP and a CNN for handwritten digit recognition. We will be training and testing both networks on the EMNIST [1] image database which was created in the same way as the well-known MNIST database which is regularly used to test various image processing systems. On this we would expect CNN to out-perform MLP as CNNs have achieved the lowest published error rates for MNIST [2] therefore we would expect our results to match this. With these low error rates, which have been near or better than human performance on the MNIST dataset, it is may even be that the existing error rate is due to misclassification of MNIST dataset. Therefore, the EMNIST which was published in 2017 may lead to different results. We will compare our results to see the relative performance of both networks on the new EMNIST dataset, and that CNN still outperforms MLP on this newer dataset and its lower error rate was not due to some specific feature of the MNIST dataset.

2. Multilayer Perceptron (MLP)

A MLP is a class of feedforward artificial neural network, consisting of at least three layers of nodes: an input layer, one or more hidden layers and an output layer. The data travels through these layers in a forward direction from input layer to output layer of neurons, and are trained using back-propagation algorithm. In back-propagation, the error is calculated using the least squares method. This works by the adjustment of the weights in relation of the size of the error compared to the expected result and using gradient descent aims to minimise the error function. The steps involved are; the network is propagated, the error is computed, then the error is back-propagated and the weights are changed.

MLP has three distinctive characteristics [3]; the neurons include a non-linear activation function, the network contains one or more layers of hidden neurons which enable the network to learn complex tasks, and that the network displays high degrees of connectivity. MLPs may suffer from over-fitting, high computational cost, and local minima can be a problem with optimisation. Unlike some other neural networks there is also a lack of biological plausibility.

3. Convolutional Neural Networks (CNN)

A convolutional neural network in a special class of artificial neural network as described above, which is a feedforward network using local connections and weight-sharing. These networks require minimal pre-processing [4]. They are based on biological processes and have been very successfully used for analysing two-dimensional shapes.

They comprise of an input layer and a series of feature maps which alternate between convolution and subsampling layers. The convolutional layers apply a convolutional operation to the input, passing the result on to the next layer, so that there are a fewer number of free parameters, allowing a deep network without the large number of parameters than would be needed otherwise in a fully connected feedforward network. The weights are the same in convolutional layers, which means the same filter can be used in each receptive field. The subsampling layers either take the largest value or average the values of one section of an image and creates a new reduced number of features maps from the previous layer. So essentially, the convolution stage creates a series of smaller feature maps while the subsampling stage reduces the number and size of these maps.

There are several benefits of these networks, mainly that they have an increased generalization ability, with an MLP of “manageable size able to learn a complex, high-dimensional, nonlinear mapping by constraining its design through the incorporation of prior knowledge about the task at hand” [3]. As

previously stated, they currently seem to provide one of the best image recognition solutions. However, in real life applications they may not perform as well as expected [5].

4. Dataset

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of labelled handwritten digits that is commonly used for testing various image processing systems, and was created from the NIST dataset in 1999 [6]. An extended dataset also created from the NIST like MNIST called EMNIST was published in 2017 and contains over 800 thousand images of both handwritten letters and numbers. This dataset was also created from the original NIST dataset using the same conversion methods, which includes shifting, blurring, scaling, etc. [1] and shares the same image structure and parameters though making better use of all available space in the image, so that this dataset can be used to allow for compatibility with classifiers already using MNIST. In addition, the performance of the EMNIST datasets outperform the original MNIST dataset at every network size. We are only using a subset of the EMNIST dataset consisting of the digit class of images which in the original NIST dataset was around 400,000 samples. In the MNIST dataset there are 70,000 digits, 60,000 training and 10,000 test samples. In the ENMIST dataset there are 240,000 training images and 40,000 testing images of digits with classes are equally distributed [1].

To obtain preliminary results, we only trained our models on 10,000 and 1,000 training and testing images respectively. The frequencies for each digit in each dataset are displayed in the histogram below.

Figure 1. Overlays of all images in each class of the EMNIST dataset

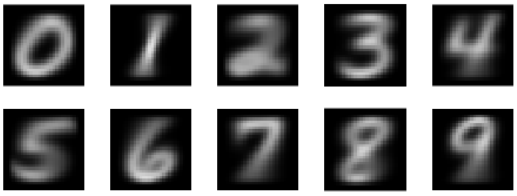
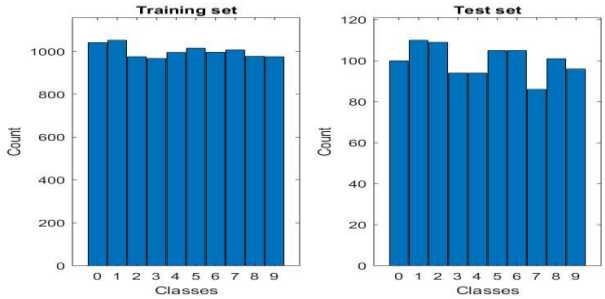


Figure 2. Distribution of digits in our initial training and test datasets



5. Hypothesis

MLP was a popular machine learning algorithm in the 1980s for image recognition, however since the development of various other neural networks and models, including CNN, they have been surpassed in performance for image classification tasks. Due to this there doesn't seem to be a benchmark MNIST score available. By contrast CNNs have been at the forefront of MNIST classification performance with error rates reaching near human performance [2]. Therefore, we would expect CNN to significantly outperform MLP on the EMNIST dataset. We may also expect CNN to perform at similar levels on the EMNIST dataset as the MNIST dataset. However, as we only aims to compare MLP and CNN, we would not implement complex network structures, thus would not expect to replicate the top performing CNN error rates that have been achieved of around 0.23 percent error [2].

6. Methodology

6.1. Training and testing strategy

As previously stated, MLP uses the back-propagation algorithm. When training starts, weights are initially set randomly, then the aim of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs. This works by going through an initial forward pass through MLP with the default weights, then calculating the total error by calculating the error for each output neuron using the squared error function and sum them to get the total error. The goal with backpropagation is to update each of the weights in the network so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole. Then this error is back-propagated back through the network and changing the weights. Each pass through the training set is called an epoch. Typically, back-propagation takes several epochs to converge. CNNs are also trained using a back-propagation algorithm though with a reduced number of free parameters due to the convolutional operation.

The original input data as a 10,000 x 784 matrix were passed through multiple, hidden layers of neurons, with the number of layers and neurons being some of our parameters to a final output layer of 10 neurons. To create the input data for CNN, we reshaped the original images which were represented as a vector with length of 784 to 28x28 pixel images. We then, as discussed earlier, used a series of convolutional and pooling layers between the 28x28 image and the output layer of 10 neurons. The number of layers, stride and frames again were used as parameters. Both networks were optimised using the Stochastic gradient descent with momentum method, with details specified in Table 1 below.

Table 1 – Details of the optimisation method used

Method	Mini-batch gradient descent with momentum
Initial learning rate	0.001
Mini-batch size	100
Number of epochs	10
Validation frequency	Each 50 iterations
Regularisation	L2, value = 0.0001
Momentum factor	0.9

In addition, the training data were shuffled at each epoch to address overfitting [7]. Initially we trained CNN and MLP on a reduced ENMIST dataset of 10,000 images to obtain preliminary results.
The models were trained on a Dell Inspiron 15 laptop with 4GB DDR3 Ram, Intel Core i5-5200U CPU, NVIDIA GeForce 920M GPU with 3.5 computing capability.

We tested MLP and CNN on 1,000 test images to compare the performance for both networks. We choose not to use a validation dataset as we were not using large set of different parameters, therefore we compared the different accuracy rates on the test data using the different parameter choices to select the best model.

6.2 Choice of parameters

We chose to use several parameters to adjust to find the best performance. The main parameter choice was around the number of hidden layers, the number of nodes in each layer, and activation function. For both models, we implemented 2 variations to the original networks, one with an extra layer, and one with an activation function at each layer. The intention is to understand the impact of activation function in comparison with an extra layer to the networks.

6.2.1. Multilayer perceptron

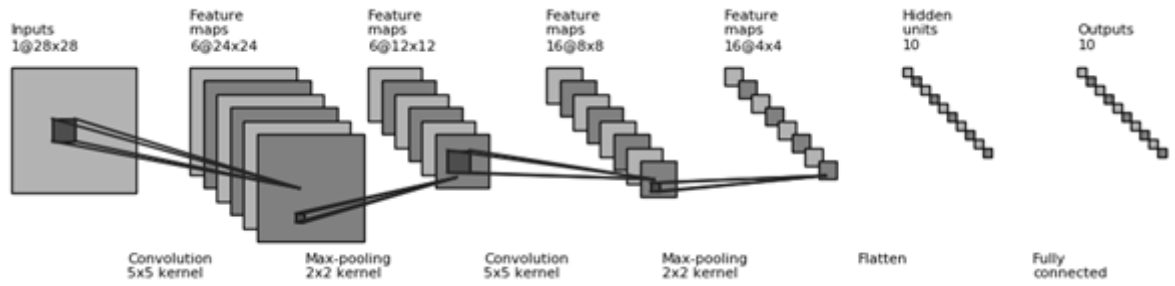
For our initial training run, we used three fully connected hidden layers of 512, 512 and 256 nodes without any activation function. We chose 512 and 256 as the numbers of nodes as a starting point because empirical results suggested that the number of nodes should be approximately equals to two-third the length of the flattened input data [8]. We choose to use a rectifier linear unit which is a very popular activation function for deep neural networks [9] and has been show to enable better training of deeper networks. These rectified linear units (ReLU) are often used in computer vision [10].
Our variations regarding to MLP are as follow:

- Variation 1: Add one fully connected hidden layer of 128 nodes after the third hidden layer.
- Variation 2: Implement ReLU activation function on the output of each hidden layer.

6.2.2. Convolutional neural network

The architecture of the original network is specified in figure 3:

Figure 3: The initial CNN implemented with 2 convolutional-max pooling blocks, followed by 1 hidden layer.



Our variations regarding CNN are as follow:

- Variation 1: Add one 2x2x32 Convolutional Layer with same padding followed by a 2x2 Max Pooling Layer before the fully connected layer.
- Variation 2: Implement ReLU activation function on the output of each Convolutional Layer.

Based on the validation accuracy, we then select the best MLP and CNN models as representations for the algorithms. We then re-trained them on different training set size, be 1,000, 5,000, and 20,000 to observe the impact of the amount of training data on classifiers' performance.

7. Results

7.1 Model Selection

The result obtained by running all the above-specified models are presented in Table 2. Based on the results, it appears that ReLU is a computing efficient option and guarantees improvement on classifiers' performance. With ReLU, MLP experienced an increase in accuracy from 90.7% to 96.0%, while for CNN, the increment was from 96.4% to 98.50%. This is in line with the literature which provide an explanation to such impact is that ReLU activation function introduces non-linearity to the models while allowing deep neural networks to be trained by addressing the vanishing gradient problem.

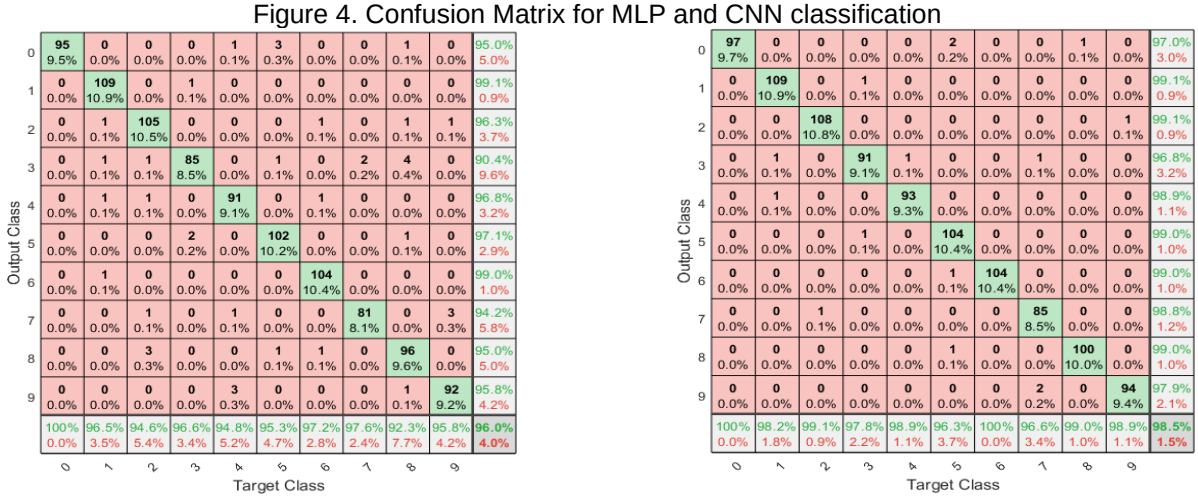
Table 2: Training and validation accuracy for the different model variations

Models	Training time (s)	Training accuracy	Validation accuracy
1. Original MLP	46	94.27%	90.70%
2. Original MLP plus 1 FC layer with 128 nodes	51	94.18%	91.30%
3. Original MLP with ReLU activation function	48	99.88%	96.00%
4. Original CNN	64	99.00%	96.40%
5. Original CNN plus 1 convolutional and pooling layer	76	98.58%	97.20%
6. Original CNN with ReLU activation function	72	99.88%	98.50%

7.2. Algorithm Comparison

Figure 4 represents the confusion matrices output by the best performing MLP and CNN. We can see a slight under-performance by MLP in classifying 3 and 8, which is reasonable given the similarity between these 2 numbers. However, CNN performed equally well in all classes.

Figure 4. Confusion Matrix for MLP and CNN classification



MLP Confusion matrix

CNN Confusion matrix

Figure 5 shows some of the misclassifications by MLP on the left and CNN on the right. "T" represents the given label and "P" which digit MLP or CNN has categorised the image as. As we can see some of these images can be tricky even for humans to categorise especially those in misclassified by CNN.

Figure 5: Misclassifications by MLP (Left) and CNN (Right)

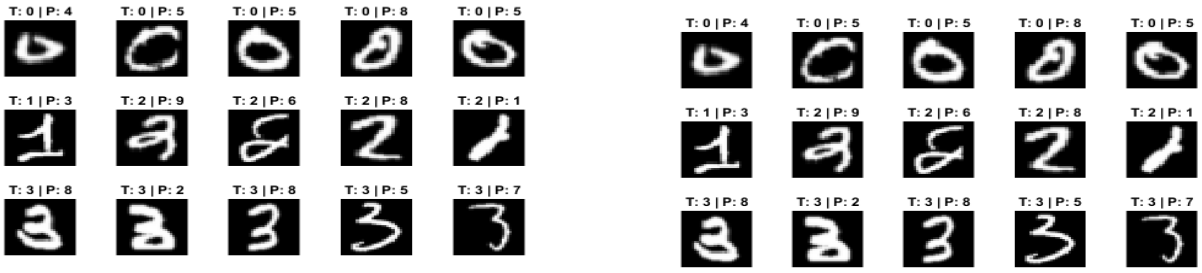
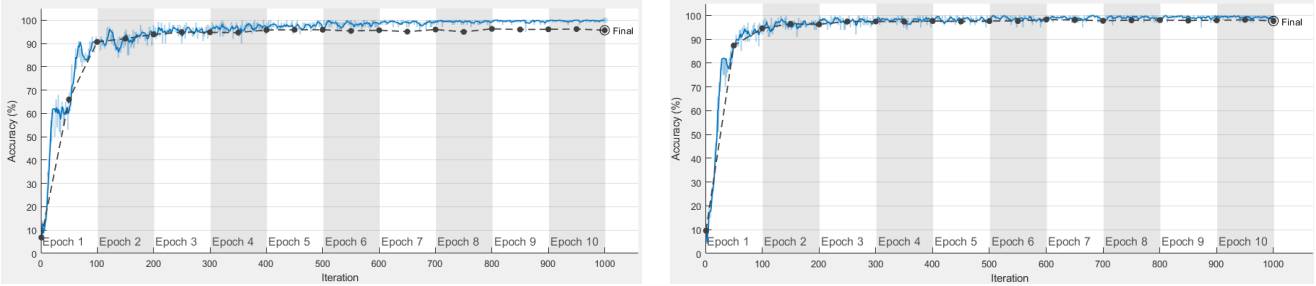


Figure 6 represent the recorded accuracy and loss of MLP and CNN during the training progress. We can see that the loss decreased quickly in the first epoch and slowly improved in later epochs. However, CNN initially learns the patterns in the data more quickly than MLP. In addition, MLP slightly overfits the data while CNN performs equally well on both training and testing data. In the next section, it is shown that MLP benefits more from increasingly large amount of training data than CNN does.

Figure 6 – MLP training progress (Left) and CNN training progress (Right)



7.3. Further results and discussions

7.3.1. Varying training set size

We tested the best models selected from above with varied training dataset size. The results are displayed in Table 3.

Table 3: Effect of dataset size on accuracy

Dataset size	MLP accuracy		CNN accuracy	
	Training	Validation	Training	Validation
1.000	93.80%	88.80%	99.70%	95.30%
5.000	99.62%	94.70%	99.64%	97.00%
10.000	99.88%	96.00%	99.37%	98.00%
20.000	99.98%	97.80%	99.47%	98.10%

The results suggested that both MLP and CNN's accuracy was increased with the size of the training dataset. CNN initially has much more success than MLP, a reasonable explanation being that the architecture with convolutional and pooling layers allows CNN to preserve the shape of the images and perform learning. However, this is not achievable by MLP as the input data is flattened at the beginning. CNN's performance seems to saturated after 10,000 training examples, so it may not be necessary to train it on a larger training set to obtain higher accuracy. However, MLP clearly benefits from an increase in the size of the training dataset as overfitting is reduced and accuracy increases.

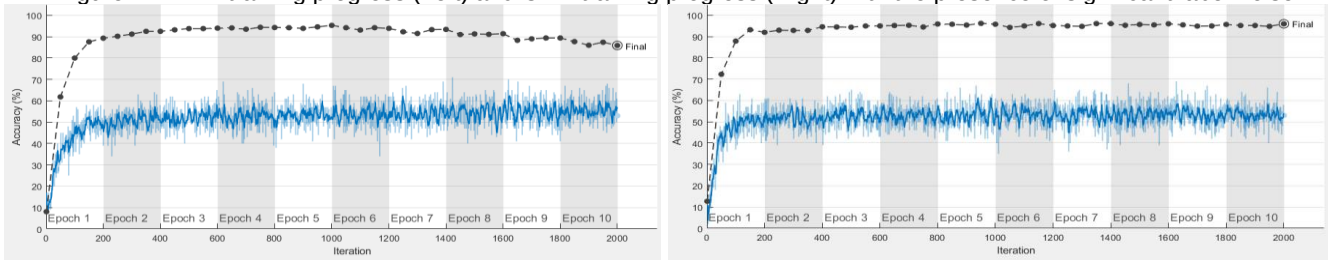
7.3.2. Testing robustness against incorrect labels

We tested the robustness of both algorithms by training the selected best performing models on a dataset of 20,000 images, half of which are incorrectly labelled. The results in table 4 show that both are able to generalise extremely well on unseen data regardless of label noise. Interestingly, Figure 6 suggests that MLP could have clearly benefitted from early stopping as the validation accuracy peaked at 95% at the end of Epoch 5. In addition, the test set performance obtained is slightly lower than if the model was trained on correctly labelled data (1% for MLP – with early stopping, and 2% for CNN). This is in line with recent empirical results [11].

Table 4: Performance against label noise

MLP accuracy		CNN accuracy	
Training	Validation	Training	Validation
56.78%	85.90%	53.64%	96.00%

Figure 7 – MLP training progress (Left) and CNN training progress (Right) with the presence of significant label noise



Regarding the dataset, EMNIST was created by using a variety of methods, such as scaling shifting rotating. This could make the data more challenging to be predicted using models trained on other simpler digit datasets. However, for classifiers that are trained on the EMNIST dataset, the performance would likely be positive, as such image altering operations can be considered as data augmentation method, which is one of the main factors behind the success of AlexNet and VGGNet [12] [13].

There have been critics about how neural networks learn, is that they tend to memorise the input data [14], which means that they will perform extremely well trained on a very large dataset. In such scenario,

there is a chance that the images used for testing the classifier would be similar to one the classifier has seen before. This maybe the case of MLP where it treats every image as an array of number [15]. However, as discussed above, convolutional layers allow CNN to learn extract features such as edges, corners from the images, which benefits learning speed and generalisation [16].

8. Conclusion and future works

By using a variety of parameters, both models greatly increased their accuracy rates, with MLP improving from 90.70% accuracy to 96% accuracy, and CNN improving from 96.4% to 98.50%. This showed the importance of different variations on the models to maximise accuracy, especially the activation function. Implementing the activation function introduced non-linearity into the performance of the models so this was a useful lesson in the difference in performance of models with different activation functions. In addition, both algorithms performed better with additional training data, especially MLP. Interestingly, both algorithms are surprisingly robust against massive label noise.

This was a relatively simple dataset only consisting of 10 classes therefore further work could be to also include the letter images in the ENMIST dataset. It would also be interesting to increase the size of the training dataset to see if there is a stage when the increases in accuracy will level out. As the best performance on the MNIST dataset is 0.23% error we could also look to maximise the performance by adapting other parameters such as the momentum, adding extra layers, implementing other regularisation techniques such as drop out and batch normalisation. It may also be useful to compare the performance of the models trained on the EMNIST dataset with a test dataset from the MNIST dataset to see the differences in accuracy.

References

[1] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “EMNIST: Extending MNIST to handwritten letters,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2017–May, pp. 2921–2926, 2017.

[2] D. Cireşan, U. Meier, and J. Schmidhuber, “Multi-column Deep Neural Networks for Image Classification,” Feb. 2012.

[3] S. Haykin, “Neural networks : a comprehensive foundation,” 2nd ed., Upper Saddle River, N.J.: Prentice Hall, 1999, p. 157.

[4] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *Handb. brain theory neural networks*, vol. 3361, no. April 2016, pp. 255–258, 1995.

[5] P. Ahmadvand, R. Ebrahimpour, and P. Ahmadvand, “How popular CNNs perform in real applications of face recognition,” *24th Telecommun. Forum, TELFOR 2016*, no. October, 2017.

[6] Y. LeCun, C. Cortes, and B. C, “MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.” [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 27-Feb-2018].

[7] Q. Meng, W. Chen, Y. Wang, Z.-M. Ma, and T.-Y. Liu, “Convergence Analysis of Distributed Stochastic Gradient Descent with Shuffling,” Sep. 2017.

[8] J. Heaton, “Introduction to neural networks with Java,” Heaton Research, Inc., 2008, p. pp 214-215.

[9] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” *AISTATS ’11 Proc. 14th Int. Conf. Artif. Intell. Stat.*, vol. 15, pp. 315–323, 2011.

[10] P. Ramachandran, B. Zoph, and Q. V. Le, “Searching for Activation Functions,” pp. 1–13, 2017.

[11] D. Rolnick, A. Veit, S. Belongie, and N. Shavit, “Deep Learning is Robust to Massive Label Noise,” 2017.

[12] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” pp. 1–14, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[14] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” 2016.

[15] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, Mar. 2007.

[16] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014.