Normalized Determinant Pooling Layer in CNNs for Multi-Label Classification

Alessandro Giuliano*[a], Waleed Hilal[a], Naseem Alsadi[a], John Yawney[b], S. Andrew Gadsden[a]
[a]Intelligent and Cognitive Engineering (ICE) Lab, Hamilton, ON L8S 4L8, Canada;
[b]Adastra Corporation, Toronto, ON M5J 2J2, Canada

ABSTRACT

Convolutional neural networks (CNNs) are a widely researched neural network architecture that has demonstrated exemplary performance in image processing tasks and applications compared to other popular deep learning and machine learning methods resulting in state-of-the-art performance in many image processing tasks such as image classification and segmentation. CNNs operate on the principle of automated learning of filters or kernels in contrast with hand-crafted digital filters to extrapolate features from images effectively. This paper aims to investigate whether a matrix's determinant can be used to preserve information in CNN convolutional layers. Geometrically the absolute value of the determinant is defined as a scaling factor of the linear transformation resulting from matrix multiplication. When an image's size is reduced into a feature space through a convolutional layer of a CNN, some information is lost. The intuition is that the scaling factor that results from the determinant of the pooling layer matrix can enhance the feature space introducing scaling as a piece of information in the feature space as well as lost relations between adjacent pixels.
Keywords: Convolutional Neural Networks, Computer Vision, Determinant, Pooling.

1. INTRODUCTION

In the past few years, the advent of machine learning and deep learning has provided a good performance on a variety of problems that span from image classification to natural language processing and forecasting. The increase in computational capabilities achieved through the use of graphical processing units (GPUs) for training and newly formed architectures have pushed the envelope in such tasks. In particular computer vision has received widespread attention in the literature, increasing the performance of models on image-related problems such as image classification, object recognition, and image segmentation. The modern framework of CNNs developed from the work of Fukushima et al. [1] on Necognitron, developed in 1980 and later refined by Lecun *et al.* through the multi-layer neural network called Le-Net for handwritten digits recognition and document recognition [2] [3]. Since then, the technology has been refined to overcome various issues encountered during training, most notably AlexNet [4] was a stepping stone in the development of deeper neural network models that can achieve very high accuracy on various image processing tasks. Krizhevsky *et al.* paved the way for CNN architectures with deep structures and multiple layers leading to the creation of ZFNet [5], GoogleNet [6], and ResNet [7]. The increase in model dimension and depth has allowed these models to get better feature representation and extrapolation, however, also resulting in greater complexity and making them more prone to overfitting.

Convolutional neural networks are made of blocks of different dimensions composed of a convolutional layer, a pooling layer, and an activation layer repeated over to create deeper structures. The performance of a given model is influenced not only by its structure and therefore the type and dimension of layers used but also by the type of loss function, regularization, and optimization used. In particular, the pooling layer has been shown to have a significant effect on model performance. There are two main pooling categories, local and global pooling, where local pooling preserves the channel dimension while global pooling aggregates the maximum of all channels to one. Within these categories, the most used pooling functions are average and maximum pooling which respectively take the average and the maximum of each pool. This paper proposes a new pooling function, based on max pooling that aims to enhance the performance of CNN on image processing by leveraging the determinant of the pools to preserve information lost during the pooling operation. The maximum pooling function inherently loses the relations between adjacent pixels while the proposed normalized determinant pooling function preserves image segment features resulting in more defined borders within the image and increasing classification performance. The performance of the proposed pooling layer was benchmarked on the same two CNN architectures across multiple multi-label classification datasets such as MNIST [3], FashionMNIST [8], and EuroSAT [9].

Computational Imaging VII, edited by Jonathan C. Petruccelli, Chrysanthe Preza,
Proc. of SPIE Vol. 12523, 1252309 · © 2023 SPIE
0277-786X · doi: 10.1117/12.2663916

2. CONVOLUTIONAL NEURAL NETWORKS

CNNs layers differ from regular fully connected neural networks in the relationships between hidden layers. Contrary to a multilayer perceptron (MLP) the convolutional layers only connect to a subset of the neurons present in the previous layer which allows for better feature extraction, isolating areas, and edges of images. Using learnable kernels, the images are convoluted producing feature maps of various dimensions depending on model size [10]. Each neuron map is referred to as the neuron's receptive field and is the result of the convolution followed by an activation function such as a rectifier, sigmoid, or tanh [2]. There exist multiple types of convolutions such as tiled convolution, dilated convolution, transposed convolution, and depth-wise separable convolution, which fall outside the scope of this article in which the focus will be placed on regular convolution and pooling functions [2], [11]. The overall architecture of CNNs uses convolutional blocks to extract the feature maps which are then projected and flattened to fully connected layers that are used to output the model prediction in the case of classification tasks.

2.1 Convolutional Layers

CNNs are capable of higher accuracy and more efficient feature extraction thanks to the convolutional layers present in the models. Contrary to a general fully connected artificial neural network the convolutional layer is only connected to a small region of the input volume, called receptive field size, which reduces the number of neurons in the overall architecture and allows the model to focus on salient parts extracting relevant features. For example, if we have an image of size 32x32x3 a fully connected network will have 3072 learnable parameters for processing the input while if we use a convolutional layer with a receptive field of 5x5 there will only be only 75 weights in that layer, reducing the complexity through the optimization of its structure [12].

The convolutional layer, therefore, slides kernels of given sizes across the input image to extract features, training the kernel weights to act as filters which are usually of small spatial dimensions. Convolution can be used for inputs of various dimensions, for 2D images the convolutions can be described as a form of template matching or feature detection [11]. The feature maps are obtained through element-wise matrix multiplication of the pool and the kernel, where the pool is a subsection of the original input. The kernel is therefore slid throughout the image depending on the dimensionality of the original input, of the kernel, and on the stride selected. The stride is the scalar value associated with the sliding of the kernel in terms of the distance between the previous and current position of the pool element covered. A visual representation of the sliding of the kernel across and input can be seen in Figure 1, the result of the convolution, also referred to as cross-correlation in the case of symmetric weight vectors, is the element-wise matrix multiplication of the pool and the kernel.

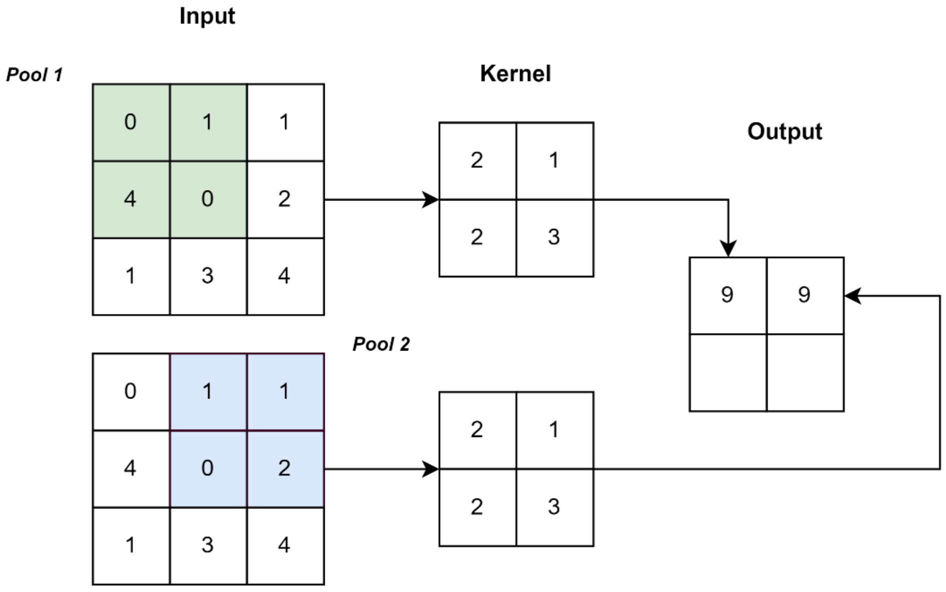

Figure 1. Visual representation of Kernel operation in convolutional layers.

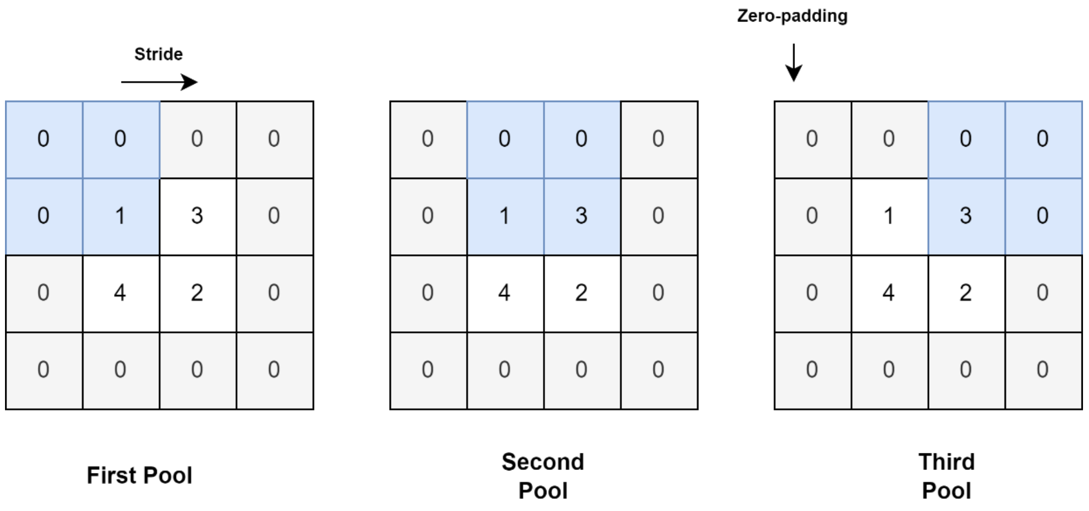

Figure 2. Visual representation of padding and stride in convolutional neural networks.

Mathematically the convolution weights or kernel parameters can be evaluated in a two-dimensional convolution as follows:

$$Y_{i,j} = [W * X](i,j) = \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} w_{u,v} x_{i+u,j+v} \tag{1}$$

where $W \in R^{H\,x\,W}$ is the filter, $X \in R^{n\,x\,m}$ is the input patch centered at (i,j) of the layer [11].

The output of the kernel operation is usually passed through an activation function that introduces nonlinearities, which is a desirable feature for the subsequent layers. Typical activation functions are ReLU, sigmoid, and tanh, and take the mathematical form:

$$z_{i,j} = a(Y_{i,j}) \tag{2}$$

Another component of the convolutional layers is padding. Zero-padding is the process of adding a border of zeros around the input image to give further control of the dimensionality of the output volume, which makes the output the same size as the input and takes the name of the same convolution, a padded convolution can be seen in Figure 2. Padding is calculated mathematically for an input size of (Height, Width, Depth), kernel size *R,* size of zero-padding *Z*, and stride *S* as follows [12]:

$$\frac{(H * W * D) - R + 2Z}{S + 1} \tag{3}$$

Multiple convolutional layers can be used to hierarchically detect low-level features in the first layers such as curves, borders, and edges to then extract higher-level abstractions in the subsequent stacked layers.

2.2 Pooling Layers

After the convolutional layer, a pooling layer is used to gradually reduce the dimensions of the representations, increasing the efficiency by further reducing the computational complexity. Pooling operations can be separated in global and local pooling where the former pools the value across every channel reducing the dimension of the image to a single

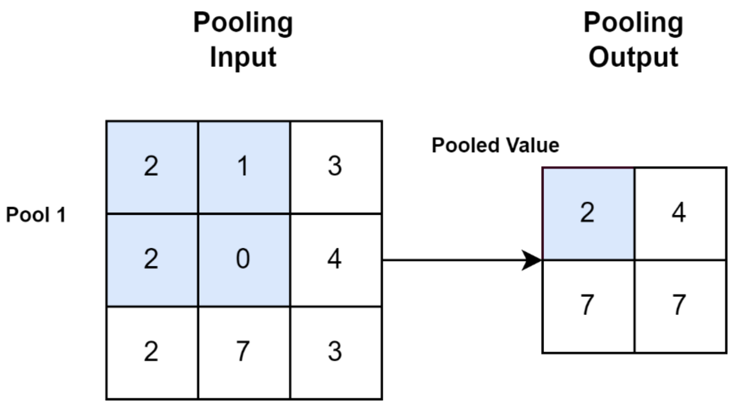

Figure 3. Visual representation of max pooling function.

channel containing the scalar values representing the feature vectors of each feature across the feature map. While the latter maintains the channel dimensionality, pooling locally each channel based on the function chosen. The most used pooling methods are maximum and average pooling, which respectively take the maximum and the average of each pool returning a scalar value and placing it in the subsequent matrix of reduced dimension. A visual representation of maximum pooling can be seen in Figure 3. Pooling functions are used to gradually reduce the input's dimension while preserving key information within the input image. The reduction in information processed by the subsequent layers addresses the problem of overfitting the training data by only focusing on salient pixels, with the tradeoff of losing some information. The downside of maximum and average pooling is that while increasing the efficiency of the information processed by focusing the attention on important regions of the image such as edges and curves, the relationship between adjacent pixels is inherently lost in the pooling operation [13]. The maximum value in each pool only represents local information, often overlooking relationships within the window.

A pooling function proposed to alleviate some of the information loss encountered in maximum and average pooling is spectral pooling [13]. The idea is to train the filters in the frequency domain parametrizing the weights as maps of complex numbers. Spectral pooling preserves considerably more information when compared to maximum pooling for the same number of parameters, the truncation and reduction of dimensionality act as a low-pass filter exploiting the non-uniformity of the spectral density [13]. Furthermore, spectral pooling allows better control of output dimensionality allowing for gradual map size reduction as a function of the layer and has shown faster convergence and optimization procedures during training. Other pooling methods include gated pooling, stochastic pooling, tree pooling, and S3 pooling [10].

2.3 Fully Connected Layers

The last layer in a CNNs architecture is the MLP layer or fully connected layer. This layer flattens in the last convolutional block output and maps it in full to a linear layer. In multiclass classification problems this last layer will have the last dimension equal to the number of classes in the problem followed by a softmax activation function to normalize the linear layer output into a probability distribution with the following properties:

$$p(y = c|x;\theta) = Softmax(f(x,\theta) \tag{4}$$

$$0 \leq Softmax(a)_c \leq 1$$

$$\sum_{c=1}^{C} Softmax(a)_c = 1$$

$$X \in R^{C} \rightarrow [0,1]^{C}$$

3. METHODOLOGY

A new pooling method is introduced in this article, offering an alternative to common pooling methods to enhance performance by decreasing the information loss derived by dimensionality reduction. To encode the dimensionality reduction within the pooling a normalized determinant function was used to scale the result of a maximum pooling operation. The determinant of pixel entries conceptually retains the diagonal difference between pixels of the pools in the pooling step. As result of the enhanced maximum pooling function outputs a reduced image with higher values for the edges and borders while also retaining some of the background information, in contrast with the normal maximum pooling which completely flattens the background. To test the proposed pooling method a simple CNN model was implemented and tested on three datasets to benchmark its performance against conventional pooling methods. In this article, a multilabel classification problem was used to determine if the function could perform equally or better than the conventional maximum pooling function.

3.1 Normalized Determinant Pooling

The normalized determinant pooling layer is implemented as a dimensionality reduction layer after the convolutional layer to better extract key features from the input image, creating related feature maps to enhance the training. Mathematically the pooling function is implemented as a scaling normalizing constant of the pool determinant multiplied by the result of a maximum pooling function and then added to the maximum pooling function itself:

$$Y_{i,j} = maxpool(X_{i,j}) + maxpool(X_{i,j}) \cdot e^{-\frac{Det(X_{i,j})^2}{2}} \tag{5}$$

Where $X_{i,j}$ is the input pool, $Y_{i,j}$ is the reduced output of j pool in image i, max pool is a function returning the highest value in a pool as shown in Figure 3, and Det is the function returning the determinant of the convolution result. For a 2x2 kernel, the determinant is expressed as follows:

$$Det(X_{i,j}) = ad - bc \tag{6}$$

$$X_{i,j} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The function's output can be visualized as a filter, Figures 4 and 5 show the result of a given image reduced by the function. It is clear when representing the function as an image filter that a lot more information is preserved compared to maximum pooling. In Figure 4 the shape of the petal is retained while also reducing the image and focusing on key aspects. In Figure 5 much of the shape of the lion's body is preserved as well as an overall shape of the background, contrary to the maximum pooling results which completely flatten out the background and lose much of the shape of the animal's body.

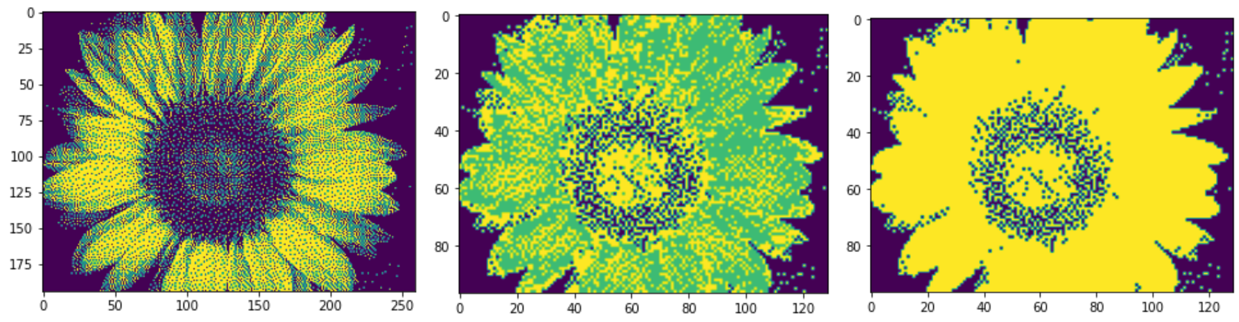

Figure 4. Representation of determinant pooling layer applied to flower image.

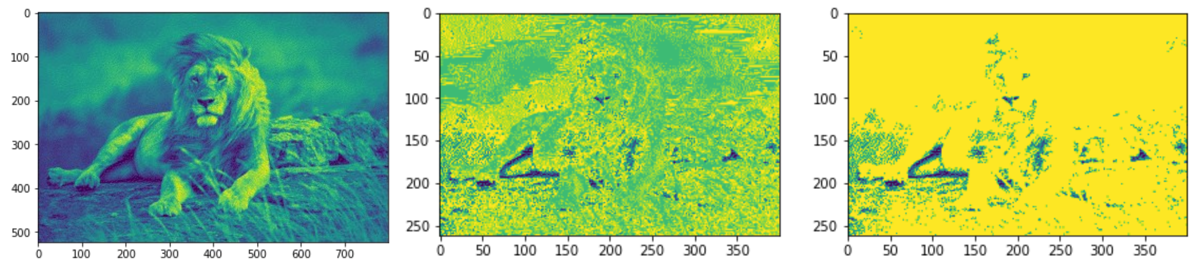
Figure 5. Representation of determinant pooling layer applied to lion image.

3.2 CNN Model

The convolutional model used to test the proposed pooling method is composed of two convolutional layers followed by pooling and relu activation functions, the output of the last convolutional block is then flattened and passed through two linear layers, culminating in a softmax activation function that outputs the probability of each class. The proposed pooling method was implemented through a custom function that fits within the PyTorch framework to optimize the process. Stochastic gradient descent was used as the optimizer for the model and tested at variable learning rates and hyperparameters to find the best combination. The loss used to evaluate the performance of the model was the negative log-likelihood loss, which is a very common loss for multiclass classification problems. The structure of the CNN model can be seen in Figure 6. Note that the batch size used for training was 64 images.

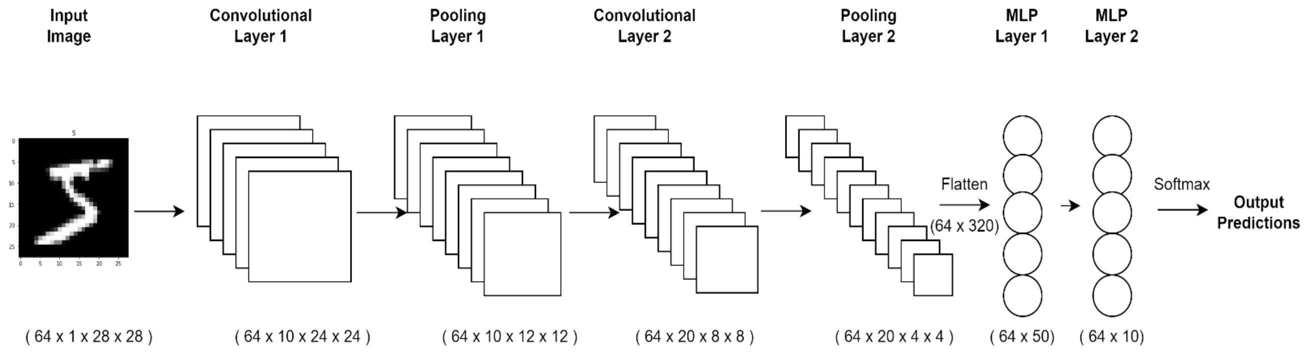

Figure 6. Overview of the model used for the MNIST dataset, dimensions vary depending on input size.

4. RESULTS AND DISCUSSION

To benchmark the performance of the proposed pooling function a multilabel classification experiment was carried out on three datasets, MNIST, Fashion MNIST, and EuroSAT. The datasets were chosen to assess the validity of the new pooling function and its effectiveness in terms of possible advantage in classification accuracy given by the preservation of information combined with the dimensionality reduction and focus on salient features. The model was run with different combinations of pooling, testing both pooling layers as determinant pooling and just the first one to be the determinant pooling layer while the second one being a classical max pooling layer. The MNIST datasets were used to assess if the function was able to outperform conventional pooling functions on simple mono-channel images while the EuroSAT was used to assess its performance on higher dimensional images with three or more channels. The tests were carried out using different learning rates to optimize the learning process as well as at multiple epochs to determine the performance at different stages of the learning process, short and long training.

Table 1. Model accuracy on MNIST dataset at multiple learning rates and training lengths.

Epochs	Learning Rate	First Pooling Function	Second Pooling Function	Accuracy
1	0.05	Max	Max	**96.9 %**
1	0.05	Det	Det	92.67 %
1	0.05	Det	Max	95.56 %
10	0.025	Max	Max	**98.66 %**
10	0.025	Det	Det	95.04 %
10	0.025	Det	Max	98.07 %

The first dataset used to test the proposed pooling function was the MNIST dataset, which contains 60,000 images of handwritten digits. The multilabel classification task is performed to recognize the written number probabilistically using softmax activation, the digits are all numbers in the set 1 to 10. As shown in Table 1, the max pooling function performed best in shorter and longer training at different learning rates. This is most likely due to the scarcity of information contained in the low-resolution images. The main point of the determinant pooling function was to preserve some of the information loss encountered by conventional pooling functions, but the results show that it is not very effective when using low-resolution mono channel images because there is not much information to preserve, and therefore a max pooling proves to be more effective.

Table 2. Model accuracy on Fashion MNIST dataset at multiple learning rates and training lengths.

Epochs	Learning Rate	First Pooling Function	Second Pooling Function	Accuracy
1	0.05	Max	Max	77.85 %
1	0.05	Det	Det	75.99 %
1	0.05	Det	Max	**78.56** %
10	0.025	Max	Max	**84.72 %**
10	0.025	Det	Det	81.13 %
10	0.025	Det	Max	83.46 %

After testing one of the simplest multi-label classification problems, the FashionMNIST dataset was used as a progressive step toward evaluating the proposed pooling function on more information-rich images. This dataset contains Zalando greyscale images of pieces of clothing, it contains 60,000 training examples, and also has 10 categories to classify from. As shown in Table 2 a combination of determinant pooling in the first layer and max pooling in the second layer performs best in the low epochs range, thanks to the preservation of information and thicker edge extraction due to the determinant pooling function. The increase in performance from the traditional max pooling function in both layers shows that there is benefits to preserving some of the key information in the pooling layers, allowing the model to converge and learn faster at equal learning rates. Although when trained for longer epochs the max pooling function still performs better, likely due to overfitting of the model to the training set. Considering that the images contained in this dataset are still mono channel low-resolution images, the tendency shown in the tests performed on the MNIST dataset by which max pooling is better performing on low-resolution images still holds when trained extensively and will most likely result in higher accuracy when the weights are trained near global optimum.

Table 3. Model accuracy on EuroSAT dataset at multiple learning rates and training lengths.

Epochs	**Learning Rate**	**First Pooling Function**	**Second Pooling Function**	**Accuracy**
1	0.05	Max	Max	32.23 %
1	0.05	Det	Det	36.15 %
1	0.05	Det	Max	**40.68 %**
20	0.1	Max	Max	58.07 %
20	0.1	Det	Det	51.3 %
20	0.1	Det	Max	**68.67 %**

To assess the performance of the proposed pooling function on multi-channel and higher resolution images the EuroSAT dataset was chosen because of the rich background information contained in the images as well as the 13 spectral bands contained in the 27,000 images dataset. This dataset was collected by the Sentinel-2 satellite and provided by the Earth observation program Copernicus. The dataset is used to classify the type of land use of the collected images such as residential, highway, and pasture, along with 7 other classes. The results in Table 3 show how the combination of determinant pooling on the first layer and max pooling on the second outperforms all other combinations by a considerable margin. In shorter training periods the proposed model is more efficient in making the model converge faster, confirming the discoveries made on the previously tested datasets, as well as performing better when trained on multiple epochs. This increase in performance is likely due to the preservation of background information from the proposed pooling function compared to the classic ones. Examples of correct labeling performed by the model can be seen in Figure 7. The max pool function flattens the background extensively causing loss of information while the proposed pooling function can retain some of that information while still reducing the dimensionality of the input and highlighting edges and borders.

The accuracy of the model could be enhanced by training the model for more epochs and by increasing the size of the model by adding more convolutional layers to extract features from the images. Further studies will be conducted to assess the performance of the proposed pooling function on larger models and trained for more epochs on the selected datasets. Furthermore, given the nature of the pooling function in preserving background information and reinforcing the edges and borders of the pooled images, it might perform best on image segmentation tasks, which will be explored in the future.

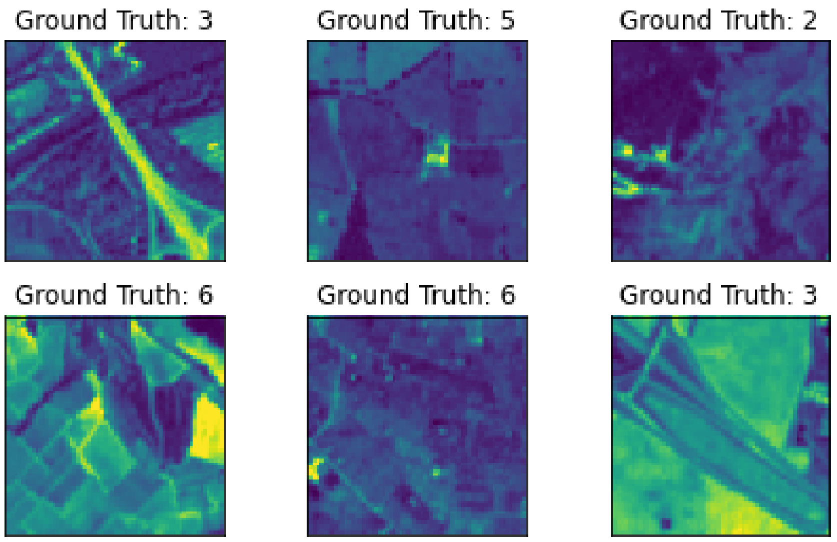

Figure 7. EuroSAT classification examples

5. CONCLUSION

A new pooling function was introduced in this article called determinant pooling, which is a modified pooling function that preserves more information during the dimensionality reduction of pooling layers. The efficacy of the newly proposed pooling method was benchmarked against three datasets MNIST, Fashion MNIST, and EuroSAT, due to the difference in their compositions. The pooling function was tested against and in combination with max pooling and showed promising results in model convergence and performance on multi-channel and information-rich images. The ability to preserve background information and to reinforce the edges and border proved very effective on the EuroSAT dataset especially due to their importance in the multi-label classification task.

REFERENCES

[1] K. Fukushima, “Biological Cybernetics Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” 1980.

[2] J. Gu *et al.*, “Recent advances in convolutional neural networks,” *Pattern Recognit*, vol. 77, pp. 354–377, May 2018, doi: 10.1016/j.patcog.2017.10.013.

[3] Y. Lecun, K. Kavukcuoglu, and C. Farabet, “Convolutional Networks and Applications in Vision.” [Online]. Available: http://www.cs.nyu.edu/

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” [Online]. Available: http://code.google.com/p/cuda-convnet/

[5] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” Nov. 2013, [Online]. Available: http://arxiv.org/abs/1311.2901

[6] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.4842

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385

[8] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.07747

[9] P. Helber, B. Bischke, A. Dengel, and D. Borth, “EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1709.00029

[10] A. Zafar *et al.*, “A Comparison of Pooling Methods for Convolutional Neural Networks,” *Applied Sciences (Switzerland)*, vol. 12, no. 17. MDPI, Sep. 01, 2022. doi: 10.3390/app12178643.

[11] K. Murphy, “Probabilistic Machine Learning.” [Online]. Available: https://mitpress.mit.edu/search-result-list/

[12] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” Nov. 2015, [Online]. Available: http://arxiv.org/abs/1511.08458

[13] O. Rippel, J. Snoek, and R. P. Adams, “Spectral Representations for Convolutional Neural Networks,” Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.03767