Universiteit Leiden
The Netherlands

Opleiding Informatica

Extending Mixed-Scale Dense CNNS

for Classification

Martijn Combé

Supervisors:
Daniël M. Pelt, Serban Vadineanu

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

13/02/2024

Abstract

Mixed-Scale Dense (MS-D) is a new convolutional neural network (CNN) architecture that makes for an easier application of deep CNNs, using dense connections and dilated convolutions instead of up- and downscaling. This ensures that all layers have the same size, which decreases the amount of parameters and the size of the network. Currently it has only been used for segmentation problems, where the objective is to obtain an image which clearly shows the different objects in the input image. The goal of this thesis was to extend MS-D to make the network useful for classification problems, where the objective is to predict to which of the pre-defined classes the object in the image belongs to. This was done by implementing different approaches and comparing them to well known classification algorithms on different datasets, in order to explore the potential benefits for using such an approach. The first approach was extending the original MS-D network with a mean operation over all pixel values and using a softmax function. The second approach for extending MS-D to classification was using a max pooling operation and three fully connected layers. It was shown that extending MS-D to classification problems using the mean-pixel approach is promising for future research, as this extension performed comparable to well-established classification networks, while using less parameters.

Contents

1 Introduction **1**

2 Background **2**

2.1 Classification . . . 2

2.2 Classification Networks . . . 2

2.3 Segmentation and Deep Convolutional Neural Networks . . . 4

2.4 Max pooling and Fully Connected Layers . . . 5

2.5 Mixed-Scale Dense networks . . . 5

3 Related Work **7**

4 Methods **8**

4.1 Method 1: Mean Pixel Value and Softmax . . . 8

4.2 Method 2: Max-pooling and Fully Connected Layers . . . 9

5 Experiments **9**

5.1 Setup . . . 9

5.2 MS-D classification experiments . . . 11

5.2.1 MNIST . . . 11

5.2.2 Fashion MNIST . . . 12

5.3 Comparison to other algorithms . . . 12

5.3.1 MNIST . . . 13

5.3.2 Fashion MNIST . . . 15

6 Discussion and Further Research **18**

7 Conclusions **19**

References **21**

1 Introduction

In the field of computer vision, many neural networks for image-processing problems exist, and each of them are designed for a specific kind of problem. Examples of such a problem is a segmentation problem, where the result is a segmented image. Here, the aim is to separate different parts within an image to increase visibility. There are many cases where segmentation can be used for real life cases, for example the segmentation of cells in medical images. Another type of problem is a classification problem, where the objective is to identify objects in an image and match them to predefined classes.
Networks for image-processing are very powerful, but use a very large amount of connections, trainable parameters and operations, which make them very computationally expensive. Mixed-scale dense (MS-D) is a neural network that is able to find more detailed solutions for segmentation problems in computer vision. This is achieved by using dilated convolutions and densely connected feature maps. [PS17] However, at this moment MS-D only exists for segmentation problems, and there is no version of MS-D for classification problems. It could be very useful if a version of MS-D exists for classification problems, where the objective is not generating an image, but providing probabilities for objects belonging to predefined classes. Classification has a lot of real life use cases, for example automation of diagnostics, face recognition and natural language processing. Prior research has shown that dense segmentation networks, like MS-D, have the ability to capture detailed information in a more effective and efficient manner compared to other segmentation networks [YK16], and that this feature is particularly useful for classification when there are limited data samples, and small-case classes [NLZJ19]. This could potentially mean that for small amounts of data and a limited amount of classes, an extended version of MS-D for classification might perform better compared to classification networks that do not use dense connections and dilated convolutions.

The goal of this research is to extend the MS-D network to classification problems, in order to explore whether the advantages of MS-D in segmentation problems still uphold when applied to classification problems. Therefore, this thesis has the following research questions:

1. How can the MS-D network be extended in order to be used for classification problems?
2. Does an extended version of the MS-D network for classification maintain its original advantages in segmentation problems when applied to classification problems?
3. How does the performance of an extended MS-D network for classification problems compare to the performance of other well-known classification networks?

In order to answer these questions, two different methods of extending the MS-D network for classification will be used. The first method uses a mean operation on all pixel values of the image and a softmax operation. The second method uses a max pooling operation and three fully connected layers. In order to investigate the potential advantages that were discovered for the segmentation version of the network, the classification MS-D will also be compared to to the classic classification networks ResNet18 and LeNet5. Popular classification problem datasets with different complexity (MNIST and Fashion MNIST) will be used to explore the capabilities of the extended classification MS-D.

This bachelor thesis is made for LIACS, and is supervised by Dr. D.M. Pelt. This chapter contains the introduction; Section 2 includes important background information for the understanding of (Deep)Convolutional Neural Networks, classification problems and the origins of the MS-D network; Section 3 Describes previous research that has been done that investigates the usage of similar segmentation networks for classification; Section 4 describes the different methods for extending the MS-D network to classification problems; Section 5 describes the experiments for comparing the different classification networks, and their outcome; Section 6 discusses the results and possible future work; Section 7 summarizes the key takeaways.

2 Background

2.1 Classification

In computer vision, image processing means taking an image as an input and extracting information. An image is defined as a set of pixels $x \in \mathbb{R}^{m \times n \times c}$ [PS17]. Here the image has m rows, n columns, and c channels. The number of channels defines the the number of colour channels of an image. The image that belongs to a single channel j of x is defined as x^j. If the image is grayscale, it has only one channel.
There are multiple types of image processing problems, one of these problems is called a classification problem. Classification problems are used in many real-life applications such as object recognition and medical diagnosis. These applications often require deep learning networks. Classification models, or classifiers, have as objective to categorize object within an image x into predefined classes C. Each image x in the dataset has a truth label y with $y \in C$. Here C is the set of possible predefined classes. The classifier function f maps an input x to a predicted class p, notated as $p = f(x)$. The accuracy of a classifier is obtained by comparing p to y.

2.2 Classification Networks

Deep learning classification networks often consist of multiple layers that have different functions. Typically, the input layer receives the raw data, and the following layers have the objective to extract features from this input using filters and convolutions. Convolutions apply a filter on the image, which selects specific features on the image, while ignoring others. A network that uses this method is often called a Convolutional Neural Network (CNN). Layer i produces a feature map $z_i \in \mathbb{R}^{m^i \times n^i \times c^i}$, which is passed on to the next layer. The features are extracted from the input image of the previous layer z_{i-1}, using a convolution. The final layer uses the feature map to calculate a probability distribution over the predefined classes.

In a common CNN structure, each layer performs multiple operations on the input image z_{i-1} before passing on the feature map[PS17]. Each channel of the input is convolved with a filter after which the different channel images are summed together. Then a constant value, called a bias, is added. Finally, a nonlinear operation like ReLu [NH10] is applied to each pixel, making it able to learn complicated functions. The mathematical notation for the output feature map z_i^j for a channel j is:

$z_i^j = \sigma(g_{ij}(z_{i-1}) + b_{ij})$.

Here, σ is the nonlinear operation and $b_{ij} \in \mathbb{R}$ is the bias. $g_{ij}(z_{i-1})$ convolves the channels of the input with a filter and sums the pixels of the image, using the following formula:

$g_{ij}(z_{i-1}) = \sum_{k=0}^{c_{i-1}} C_{h_{ijk}} z_{i-1}^k$.

Here, $C_{h_{ijk} z_{i-1}^k}$ is the two dimensional convolution with filter h_{ijk} and image z_{i-1}^k. Different alternatives for convolutional layers exist, and layers can differ from each other. One of the alternatives for a convolutional layer is a fully connected layer [SZ14]. Another alternative for a convolutional layer is the use a softmax operation [RFB15] for classification. The function for softmax is as follows:

$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

Here x_i is the value assigned to the item belonging to a class, and $\sum_j \exp(x_j)$ is the sum of the values that belong to the possible predefined classes C.
In order to achieve good results, the network has to train and optimize its parameters. The parameters determine the bias, the filters and the weight of the connections that exist between layers and can therefore determine if a feature is useful or not. The optimization of these parameters is done using a two-step plan called forward- and back propagation.
During the forward propagation step, the data is passed through the layers of the network, and class predictions are made on the training set. Then the loss is calculated to measure how well these predictions compare to the actual truth labels. One method to calculate this loss is Cross entropy loss. [PyT24b][RFB15] It uses a softmax function to assign a probability for each of the possible pre-defined classes in the range [0,1] of which all values sum to one. This results in a probability of the image belonging to each one of the classes, represented as a tensor. The resulting tensor is then used to calculate the loss with a cross entropy function. Cross entropy is a logarithmic loss function for classification. For each image, this function compares the predicted class of the network to the truth labels, and computes a loss function. This value should be as low as possible and is used to update the weights in the network using backwards propagation. The cross entropy loss L for multiple classes is as follows:

$\text{Cross\_entropy Loss}(L(y,p)) = -\sum_i y_i \times log(p_i)$

Here, p is the probability distribution over classes, and y is the ground truth label vector, where all classes are zero except the class with the true label. In the backward propagation step the parameters of the network are adjusted in order to decrease the loss, using an optimization algorithm. One commonly used optimization algorithm is the Adam algorithm [KB15]. Adam is short for adaptive moment estimation, and combines the Adaptive Gradient Algorithm (AdaGrad), known for improving performance in scenarios with a small amount of gradients, often used for natural language and computer vision problems, with Root Mean Square Propagation (RMSProp), designed to adapt effectively to dynamically changing problems [Bro21]. This unique combination makes the Adam algorithm very popular for addressing both online and non-stationary problems. The forward- and back propagation steps are repeated over multiple complete iterations of the training dataset, called epochs.

2.3 Segmentation and Deep Convolutional Neural Networks

Classification problems are not the only type of problem in the field of computer vision. Segmentation problems have the objective to create a segmented image, where different components are clearly shown by adding contrast or different colours to components. Instead of a probability distribution, which is the case for classification networks, a segmentation network therefore has an output image $\mathbb{R}^{m\times n\times c} \to \mathbb{R}^{m'\times n'\times c'}$ [PS17]. The output has the same amount of rows and columns as the original input image, but with different pixel values that indicate clearly which class a pixel belongs to.

The typical CNN described in section 2.2 is often used for simple tasks. A deep convolutional neural network (DCNN) is able to model complicated functions, by capturing features at different scales. This is often used for segmentation problems, by first downscaling and then upscaling the dimensions of feature maps. This concept is called Hierarchical Feature Extraction and is extremely useful as it combines low level information with complex high-level patterns.

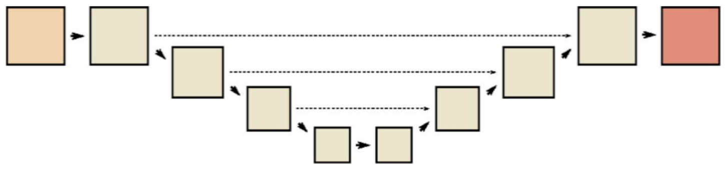

Figure 1: Structure of Deep Convolutional Neural Network U-Net, containing layers of different sizes and skip-connections. Source: Daan M. Pelt and James A. Sethian (2017)

One existing problem with DCNNs is that all layers have a different scale, making it only possible to learn from the previous layer. This is partly solved by adding skip-connections between layers that have the same size. A standard structure of a DCNN network called U-Net can be found in figure 1. Another problem is that the depth (amount of layers) of the network. DCNNs consist of millions of parameters that have to be trained. All connection between layers and layers themselves have different parameters, like size, biases, filters and weights. The large amount of parameters make it likely that training gets stuck in a local minimum, or over-fitting on the training data, which means that the networks has trained too much on the training data. This makes the network less capable of making the correct class predictions on new data. These parameters affect training time and increase the likelihood of over-fitting the network, because they exaggerates patterns that are found in the training data that might not exists in new datasets. [Bal22]

2.4 Max pooling and Fully Connected Layers

Even though DCNNs have their limitations, they are still some of the most efficient and popular networks to date. Examples of popular DCNNs and CNNs are VGG, LeNet5, Unet and ResNet18. VGG [SZ14] uses a structure in which the raw data is downsized multiple times using Maxpooling operations [Nan23] after each layer. Max pooling operations reduce the size of an input, while maintaining the most significant features and removing invariances. It uses a pooling window (kernel) and finds the most important value within that window, which is then passed on to the output image. An example of this process is shown in figure 2 [NMRW22].

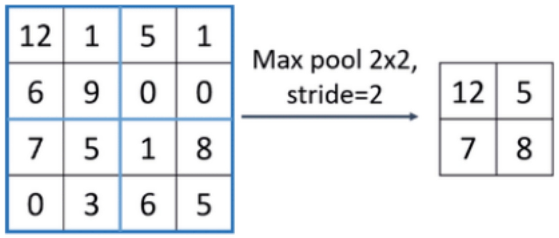

Figure 2: Example of a max pooling operation with a 2x2 kernel and a stride of 2. Source: R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang. ”Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study”

The amount of reduction in size is determined by the stride of the Max-Pooling operation. For VGG this is a small amount per convolutional layer. The formula of the Max-Pooling operation is as follows:

$$MaxPooling(X)_{i,j,k} = \max_{m,n} X_{i*s_x+m,j*s_y+n,k}$$

Here, X is the input, (i,j) are the indices of the output, k is the channel index, s_x and s_y are the stride values in the horizontal and vertical directions, respectively. m and n are indices for the pooling window. After the convolutional layers, the information of the obtained feature maps are combined in multiple fully connected layers, in which all input nodes are connected to all output nodes, which is not done in regular convolutional layers [Unz22][SZ14].

2.5 Mixed-Scale Dense networks

The disadvantage of large amounts of parameters in DCNNs that were discussed in section 2.3 are partly solved by Mixed-Scale Dense (MS-D) Networks [PS17]. This network use dilated convolutions instead of regular convolutions with up- and downscaling operations to capture features at different scales. A dilated convolution uses a dilated filter that is nonzero only at distances that are a multiple of pixels from the center. The dilation factor determines the distances between these pixels, as shown in figure 3 [DW20]. Dilated convolutions can capture additional features compared to DCNNs that use the traditional scaling approach[YK16]. The gaps in the filter ensure that large-scale information about the image becomes available faster compared to traditional methods.

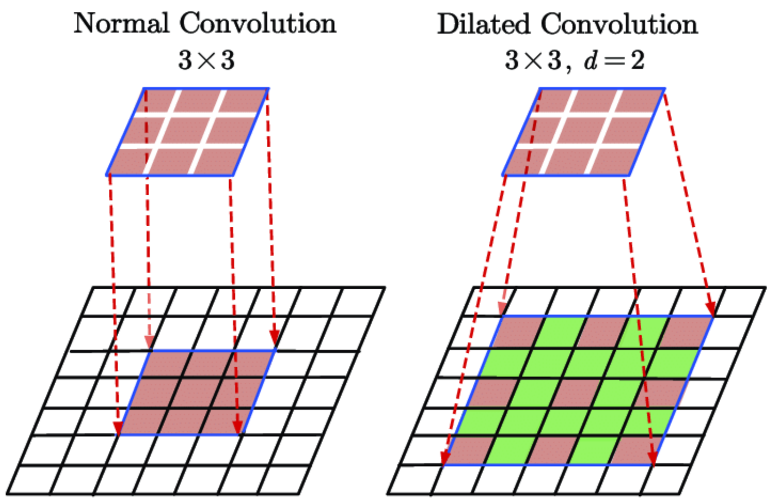

Figure 3: A representation of the difference between a normal 3x3 convolution and a Dilated convolution with a dilation factor of 2. Source: Jinglong Du and Lulu Wang. "Brain mri super-resolution using 3d dilated convolu- tional encoder–decoder network"

The change in approach compared to traditional DCNNs has an additional advantage. The scale of the different layers is not altered and therefore all input and output images have the same width and height. Because of this, there is no restriction to using only the output of the previous layer and layers of the same size. This means that all previous feature maps $\{z_0, ..., z_{i-1}\}$ can be densely connected instead of only those of the previous last layer, which is a problem that classic DCNNs can only partly solve with skip connections. This results in the following change for the output layer computation of z_i^j:

$z_i^j = \sigma(g_{ij}(\{z_0, ..., z_{i-1}\}) + b_{ij})$,

with $g_{ij}(\{z_0, ..., z_{i-1}\}) = \sum_{l=0}^{i-1} \sum_{k=0}^{c_{l-1}} D_{h_{ijkl}, s_{ij} z_l^k}$.

A schematic representation of the network, showing the dense connections of MS-D can be found in Figure 4. The dense connections also result in a decrease of parameters in the network compared to networks where each layer requires different values, which can be very useful as it makes the network less problem specific and complex. This makes it easier to use the network on new problems, since it requires less fine-tuning. Previous research has also shown that for MS-D networks, the class accuracy tends to be higher than some DCNNs, especially with a small amount of parameters. Another benefit of MS-D is that for different choices of dilations the performance of MS-D networks is similar, which makes it less difficult to fine-tune [PS17].

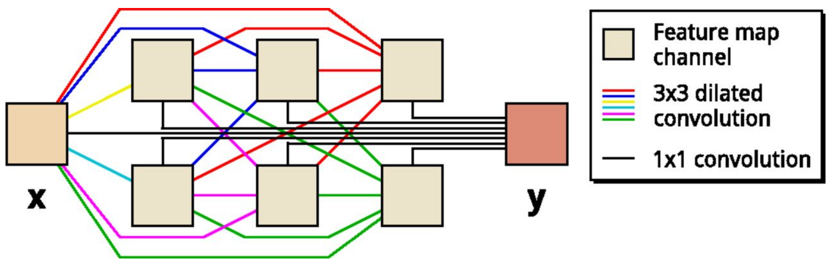

Figure 4: Schematic of the MS-D network. Source: Daan M. Pelt and James A. Sethian(2017)

3 Related Work

Previous research has been done to explore the extension from segmentation to classification of the DeepLab network[NLZJ19]. This network was originally designed for semantic segmentation tasks and was evaluated using datasets that contain multiple classes within a single image. First, the network extracts spatial features, followed by classification using a Support-Vector Machine (SVM) algorithm.
DeepLab uses dilated convolutions, a technique similar to MS-D. However, unlike MS-D, DeepLab uses separate layers for these convolutions, which are not all interconnected. It extracts features at four different rates, which are then combined and passed through a softmax function.
When classified with an SVM, this network outperforms traditional methods and existing deep learning-based networks for this kind of problem. This is mostly the case for small-scale classes with limited samples. This improvement can be attributed to the network's ability to preserve detailed information early in the process without resizing, resulting in better class separation.

Another interesting research regarding the use of dense networks for classification problems was done for the MSDNet network [Tsa19], which is not to be mistaken from the MS-D network that is used in this paper. The difference is that MSDNet is a Multi-Scale Dense network and MS-D is a mixed-Scale network. MSDNet does not use dilated convolutions, and uses two down-sampling convolutions in each layer. MS-D is more densely connected, because all feature maps have the same size. However, since MSDNet is still somewhat densely connected and is used for classification specifically, it is still interesting to view the results of this research. MSDNet uses multiple intermediate classifiers between layers to check if the classification confidence at that point in the network exceeds a certain threshold. If this is the case, the network can be stopped, to save resources. Each classifier has two down-sampling convolutional layers with 128 dimensional 3×3 filters, followed by a 2×2 average pooling layer and a final fully connected linear layer. The MSDNet research shows that networks that use dense connections outperform networks that do not. It is also shown that the intermediate classifiers work better when using dense networks, compared to implementing them in other networks like ResNet. For this reason it might be useful to implement some of the classification methods in the extended classification version of MS-D.

4 Methods

4.1 Method 1: Mean Pixel Value and Softmax

After exploring the background of DCNNs and the MS-D network and investigating the related work regarding the extension of networks from segmentation to classification problems, two methods are used for extending the original MS-D network. The first method for extending the MS-D network from segmentation to classification is to apply a mean operation to the pixel values on each image of the output layer, after the original segmentation of MS-D has finished. The mean operation [PyT24a] returns the mean value of all elements in the input tensor, which in this case are all pixels in the image. In contrast to the original version of MS-D, this version will not use dilated convolutions, but normal 3x3 convolutions without dilation. The reason for this is that dilated convolutions are mostly useful for very large images, since the method is good for capturing context at a large scale. However, the experiments of this research will use datasets with relatively small images, which removes the need for dilated convolutions. The details of the used datasets will be further explained in section 5.1. The simple MS-D implementation is still densely connected, and feature maps remain the same size. This ensures that the potential advantages of using MS-D for classification can still be explored
After the MS-D section of the network, the output layer contains the same amount of images as the amount of possible classes. Then, a softmax function is applied and cross entropy loss is used to calculate the loss. After multiple epochs, the class with the highest softmax probability will be assigned to the original image. An overview of the structure of this extension can be found in figure 5.

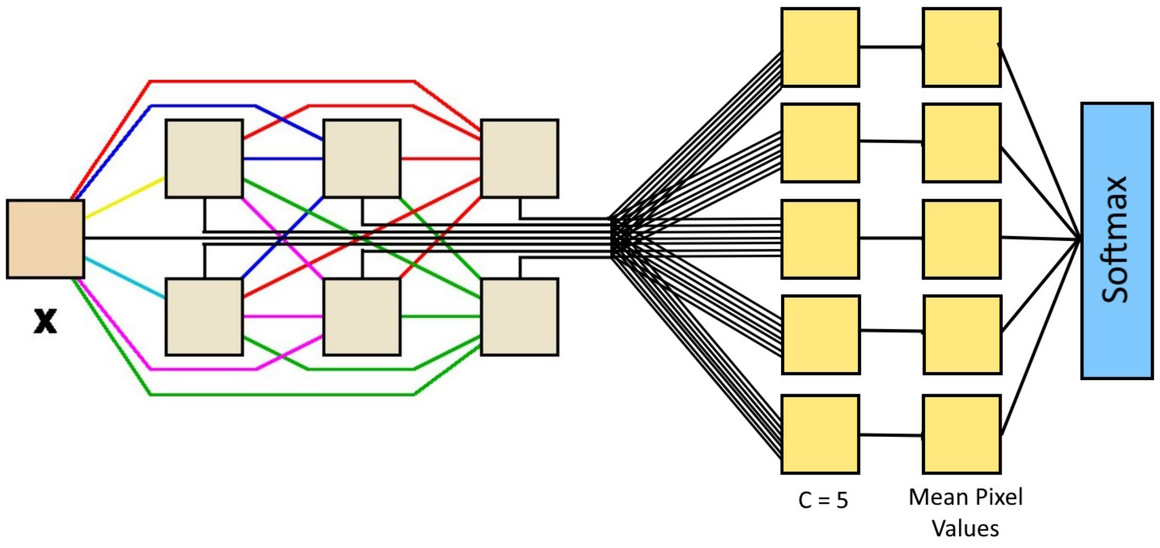

Figure 5: Schematic of the extended MS-D network for classification using a mean and softmax function, where the amount of predetermined classes(c) is equal to 5.

4.2 Method 2: Max-pooling and Fully Connected Layers

The second method that is explored takes inspiration from the popular VGG network that was previously mentioned in section 2.3. The VGG network uses multiple Max-Pooling operations after each convolutional layer that decreases the size of feature maps with a pre-determined amount of pixels (stride). Then multiple fully connected layers are used to combine the obtained information of the feature maps. For this extension Max-Pooling and fully connected layers are implemented. However, instead of performing multiple max-pooling operations, the original MS-D network without dilated convolutions is used. A single large max-pooling operation is applied in order to pass the obtained feature map to the fully connected layers. The large Max-Pooling operation reduces the image size to a size of 1x1, by simply setting the stride of the operation to the size of the original image. After the fully connected layers f1, f2 and f3, the obtained logits are passed to a softmax operation, resulting in the final prediction of the network. The first two fully connected layers have 4096 channels, which is the same amount in the original VGG16 Network [SZ14]. The final layer has as many channels as there are possible classes. A structural overview of this extension can be seen in figure 6.

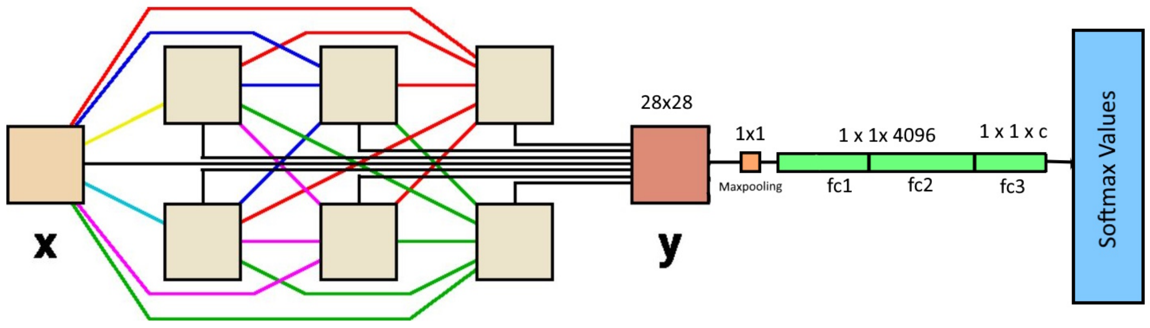

Figure 6: Schematic of the extended MS-D network for classification using a maxpool operation and three fully connected layers.

5 Experiments

5.1 Setup

The extended MS-D algorithms for classification were implemented in Python using PyTorch version 2.0.1 [PyT23]. The tests were run on PCs at LIACS, which contain a NVIDIA GeForce GTX 1050 Ti GPU, running CUDA version 12.2. For the backward propagation of the network, the Adam algorithm is used. Two different datasets are used for the experiments, MNIST [LCB10] and FashionMNIST[XRV17]. MNIST is a very simple dataset containing 60000 28x28 grayscale images of handwritten digits, as seen in figure 7. It contains 10 classes and is often used as a benchmark for classification networks. FashionMNIST contains 60000 detailed 28x28 grayscale images of pieces of clothing, as seen in figure 8.

The images of both datasets have the same size, which makes it is less complicated to implement them both in the same code. The detailed images will show how well the extended MS-D networks perform on more complicated datasets, compared to other network architectures.

All tests are conducted using a train set, a validation set and a test set of the original data. The train set contains 70 percent of the dataset (42000 images), and the test and validation set will both contain 15 percent of the dataset (9000) images. The test set will only be used to compare the performance of the different algorithms, while the validation set is used to find the best parameters of the algorithms. The different sets are produced by applying a train-test split with a fixed random seed, which will ensure reproducability. This is because all images in the test set will be exactly the same during each experiment. However, all tests are conducted three times and averaged in order to insure reliability, since the images are being fed into the network at random, which leads to slightly different results. For the experiments, two metrics are used. The first metric is class accuracy, which is a percentage of the amount of images that are classified correctly in the test or validation set. Class accuracy is a good way to measure the performance of the network, since it is the main objective of a classification network to correctly classify as many images as possible. The second metric is the average loss, which is another good way to quantify the difference between the predicted class and its true label. The average loss is obtained by using the cross entropy loss function [Ala23]. A lower average loss means that the network is better at making good predictions on the dataset.

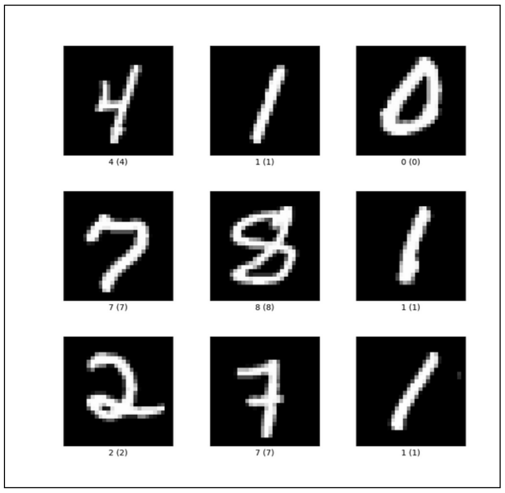

Figure 7: Example of images within the MNIST dataset.

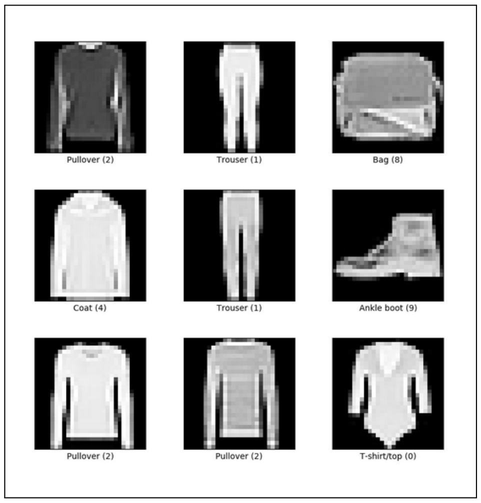

Figure 8: Example of images within the Fashion MNIST dataset.

5.2 MS-D classification experiments

The first experiment is to find the best parameters for each of the extended classification MS-D versions, so that it can be compared to more well known classification algorithms. First, the depth is varied and tested with ten epochs, and the version with the lowest loss and highest average accuracy on the validation set is kept. Then, the network is fine tuned by varying the amount of epochs. This results in the versions of the MS-D networks with the best performance on the validation set.

5.2.1 MNIST

The results of the experiment for the MNIST dataset can be found in table 1. It shows that the mean-pixel version needs a much larger depth to create reasonable results, compared to the maxpool version. At a depth of 200, the network starts to get very large and takes a long time to finish training. The version of mean-pixel with a depth of 250 has a slightly worse accuracy and has a lower loss compared to the version with a depth of 200. This shows that networks with a depth of over 200 start to over-fit on the training data. Due to hardware restrictions it is impossible to test the network beyond a depth of 250, and the training of the network takes a long time, even after a depth of 150. Therefore future experiments on the mean-pixel version of MS-D will be conducted with a depth of 200.
The maxpool version of the MS-D classification network performs much better than the mean-pixel version at smaller depths. At a depth of 50, the network already outperforms the mean-pixel version in both accuracy and average loss. This could be because the network in itself contains more layers and is more balanced with fully connected layers to combine information. Since the depth of the network affects the time to run the network, the smaller depth of the maxpool network is a large advantage. The experiments with a depth of 100, 125 and 150 yield a similar loss, and varying class accuracy's that all lie in the same range. Therefore the maxpool MS-D version with a depth of 100 will be used in the future, since it has the lowest loss.

Depth	CA mean-pixel	AL mean-pixel	CA maxpool	AL maxpool
50	94.19	0.2217	96.90	0.0809
100	96.00	0.1231	98.50	0.0460
125	-	-	98.29	0.0462
150	98.17	0.0582	98.75	0.0464
200	98.65	0.0411	-	-
250	98.63	0.0377	-	-

Table 1: The depth of the mean-pixel and maxpool version of MS-D, compared to the average class accuracy (CA) on the validation set and the average loss (AL) after training the network for 10 epochs on the MNIST dataset. Results for depths that are not relevant for finding the best version of the network are indicated with ”-”.

5.2.2 Fashion MNIST

The extended MS-D methods for classification are tested on Fashion on MNIST, in order to find if MS-D is problem specific and to see how the networks perform on more complex data compared to the MNIST dataset.

Depth	CA mean-pixel	AL mean-pixel	CA maxpool	AL maxpool
100	87.24	0.2206	86.98	0.2842
150	88.11	0.3317	89.28	0.2583
200	88.34	0.2973	87.77	0.2665
250	89.15	0.2617	-	-

Table 2: The depth of the mean-pixel and maxpool version of MS-D, compared to the average class accuracy (CA) on the validation set and the average loss (AL) after training the network for 10 epochs on the Fashion MNIST dataset. Results for depths that are not relevant for finding the best version of the network are indicated with "-".

The performance of the extended networks on the more complex dataset Fashion MNIST can be found in table 2. The results show that a larger depth is needed in order to achieve reasonable results for both extended versions. The average Loss is much larger on the Fashion MNIST dataset compared to the average loss on the MNIST dataset, since the Fashion dataset is more complex and has a difficult function to learn. This can also be observed for the average class accuracy of both extensions, which is ten percent lower, compared to the accuracy on the MNIST dataset. Both extensions have the best performance when the depth is 50 higher than on the MNIST dataset. For the mean-pixel MS-D, this means that a depth of 250 is the best option. Due to hardware constraints, this is the highest possible depth that can be tested. It is possible that an even higher depth would be more optimal. The maxpool MS-D extension needs a depth of 150. For a higher depth, this network starts overfitting, resulting is a lower class accuracy.

5.3 Comparison to other algorithms

After the optimal depth for the two extensions is found, the best versions of each classification MS-D extension are compared to the well known classification algorithms ResNet and LeNet5. This is done to set a good benchmark for the evaluation of the performance. Also, a comparison is made between the algorithms to find how much data they need to produced reasonable results. This is done by gradually decreasing the size of the train set. Because of this, the epochs are smaller. To counter this, for each network and training set percentage, the optimal amount of epochs is used, which can be found outputting the accuracy after each epoch and picking the epoch with the highest accuracy. Then, for that amount of epochs the experiment is run twice, in order to have three accuracy's that can be averaged for more valid results. As a final experiment, the amount of parameters of each model are compared to their average class accuracy, since the Mean-Pixel MS-D version uses a lot less parameters, compared to other DCNNs.

5.3.1 MNIST

Table 3 shows that there is no significant difference between the different algorithms for the MNIST dataset at training set percentages from 20 to 100. When looking at figure 9 it is clearly shown that at very small amounts of training data, Maxpool MSD significantly underperforms. Mean-Pixel MSD performs the best at this amount of data.ResNet18 performs the best at the largest amount of training data, but at lower amounts of data Mean-Pixel MSD has a higher average class accuracy. It is important to keep in mind that MNIST contains around 60,000 images, which makes it an unusually large dataset. Most datasets contain a smaller amount of images, in which case the use of the Mean-Pixel MSD network could be benefical. Maxpool MSD and LeNet5 have roughly the same performance at each training set percentage. However, figure 10 shows that the average class accuracy's do not differ a lot between algorithms, showing no significant difference in performance as the training set percentage decreases.

Method	TSP = 1	TSP = 20	TSP = 40	TSP = 60	TSP = 80	TSP = 100
mean-pixel MS-D	89.15 (75)	98.21 (40)	98.43 (20)	98.59 (15)	98.62 (10)	98.75 (10)
maxpool MS-D	83.79 (75)	97.72 (25)	97.97 (20)	98.08 (20)	98.32 (20)	98.60 (10)
ResNet18	88.38 (155)	98.10 (25)	98.27 (10)	98.44 (10)	98.56 (10)	98.91 (10)
LeNet5	88.74 (175)	97.63 (30)	97.97 (25)	98.22 (20)	98.37 (20)	98.72 (10)

Table 3: The average class accuracy on the test set of different classification networks on the MNIST dataset, using different training set percentages(TSP). The amount of epochs used is indicated between brackets.

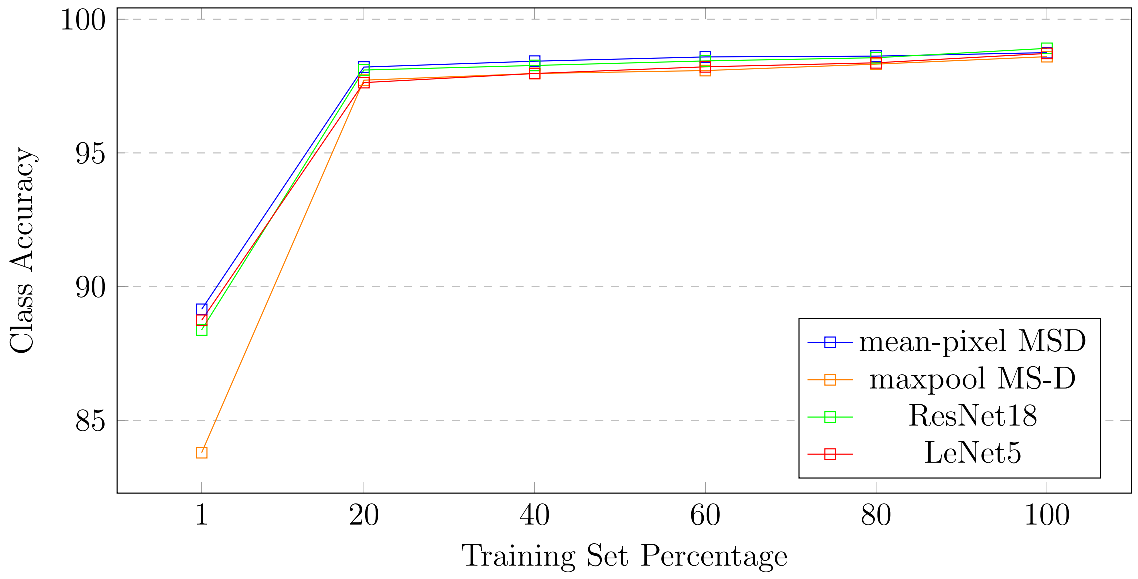

Figure 9: Graph showing the average class accuracy on the test set of different classification networks on the MNIST dataset, using different training set percentages.

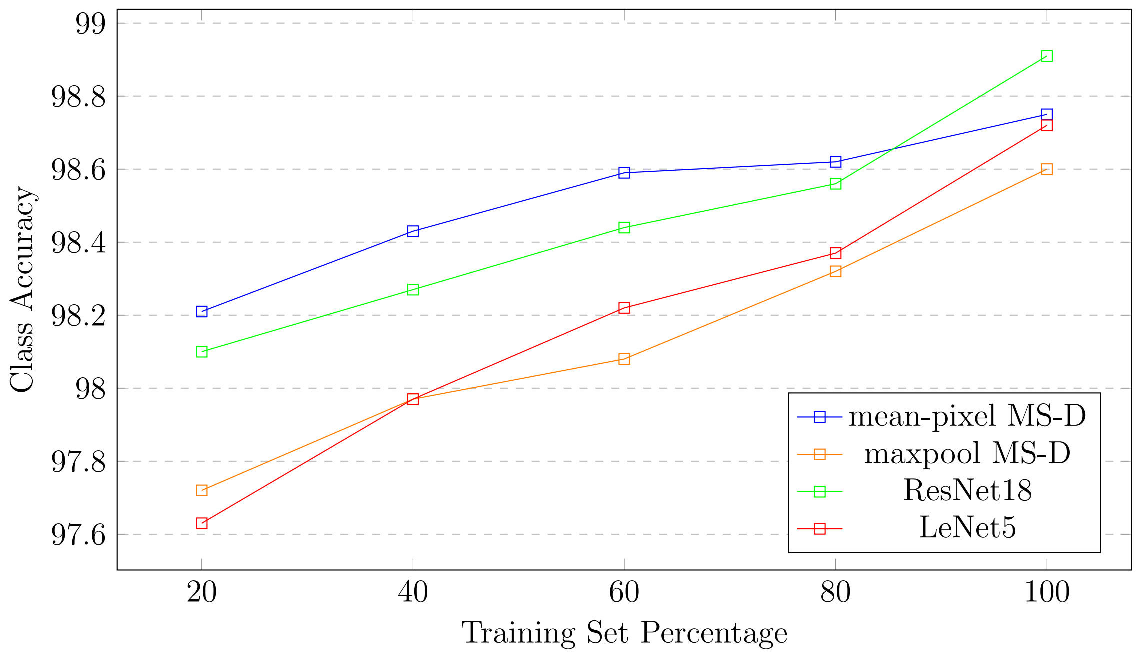

Figure 10: Graph showing the average class accuracy on the test set of different classification networks on the MNIST dataset, using different training set percentages. The figure is zoomed in on higher training set percentages in order to visualize relevant data.

The amount of parameters for the optimal version of each algorithm are compared to their accuracy. As shown in figure 11, LeNet5 and Mean-Pixel MS-D use significantly less parameters compared to ResNet18 and Maxpool MS-D. Fully connected layers use a significant amount of parameters especially if the network contains a lot of layers, because all input and output nodes are connected. This is especially the case for Maxpool-MS-D, since it requires a depth of at least 100 layers to produce reasonable results. The amount of parameters can affect training time and computational requirements. ResNet18 uses a very large amount of parameters, but has a much smaller training time compared to both the MS-D networks. This could be because ResNet18 has been optimized in a lot of ways and the MS-D networks have not. ResNet18 has a higher accuracy than LeNet5 and Mean-Pixel MS-D. However, both LeNet5 and Mean-Pixel MS-D appear to have a good balance between the amount of parameters and accuracy, resulting in high class accuracy's and in the case of LeNet5 a very fast training time. This is not the case for Mean-Pixel MS-D, since it has not been optimized yet.

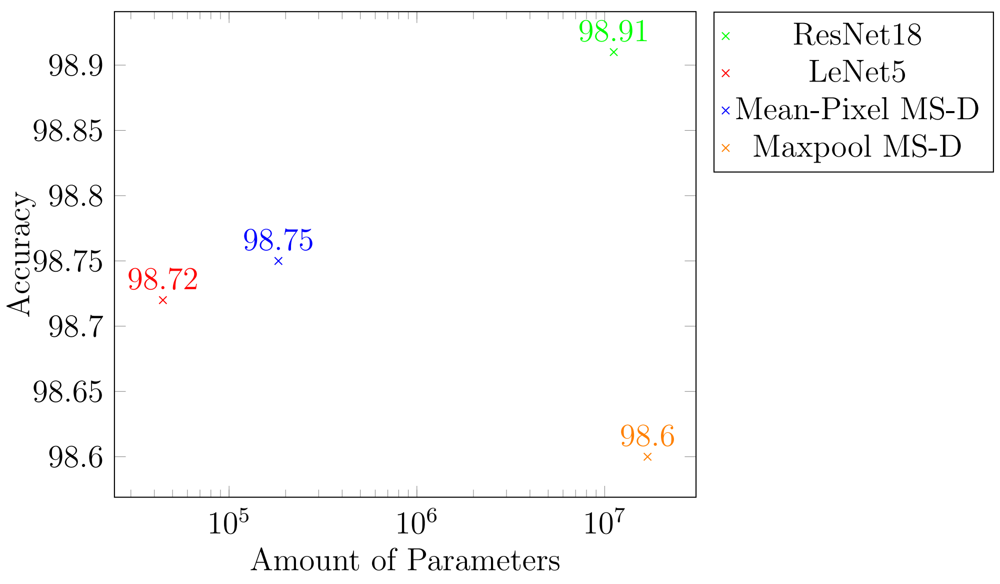

Figure 11: Scatter plot of the amount Model parameters for each of the compared networks, compared to their class accuracy on the MNIST dataset

5.3.2 Fashion MNIST

For the complex Fashion MNIST dataset, it would be expected that more epochs are needed in order to learn the more complicated function. However, as can be seen in table 4, this is not the case for both MS-D extensions. ResNet18 and LeNet5 do need more epochs to produce reasonable results. When comparing the FashionMNIST dataset results to the MNIST dataset results in table 3, the class accuracy is approximately 8 percent worse for all training set percentages. For the very small training set percentage of 1 percent, seen in figure 12, the maxpool MS-D extension performs worse compared to the other tested networks, as was the case for the MNIST dataset. On this dataset, mean-pixel MS-D performs slightly worse than ResNet18 for the smallest training set percentage. However, in figure 13, it is clearly shown that for the percentages of 20, 40, 60 and 80 the network either outperforms or performs comparably to ResNet18 and LeNet 5. This suggests that the network is able to achieve performance comparable to those of these better-known classification networks. This is not the case for the maxpool MS-D extension, since the results show that the network performs the worst on almost all training set percentages.

Method	TSP = 1	TSP = 20	TSP = 40	TSP = 60	TSP = 80	TSP = 100
mean-pixel MS-D	76.90 (75)	87.98 (40)	89.48 (15)	89.96 (15)	89.97 (15)	89.97 (10)
maxpool MS-D	70.83 (55)	86.34 (25)	87.90 (25)	88.32 (15)	89.20 (15)	89.53 (10)
ResNet18	77.54 (185)	88.03 (65)	88.54 (45)	89.07 (40)	90.14 (35)	90.78 (10)
LeNet5	75.29 (170)	86.39 (70)	89.01 (50)	89.46 (45)	89.47 (20)	89.48 (20)

Table 4: The average class accuracy on the test set of different classification networks on the Fashion MNIST dataset, using different training set percentages (TSP). The amount of epochs used is indicated between brackets.

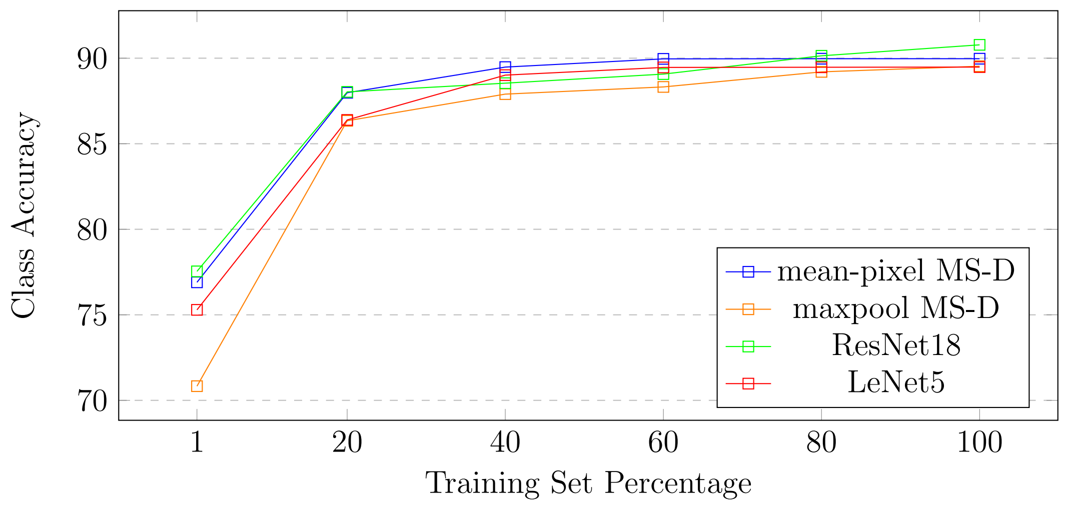

Figure 12: Graph showing the average class accuracy on the test set of different classification networks on the Fashion MNIST dataset, using different training set percentages.

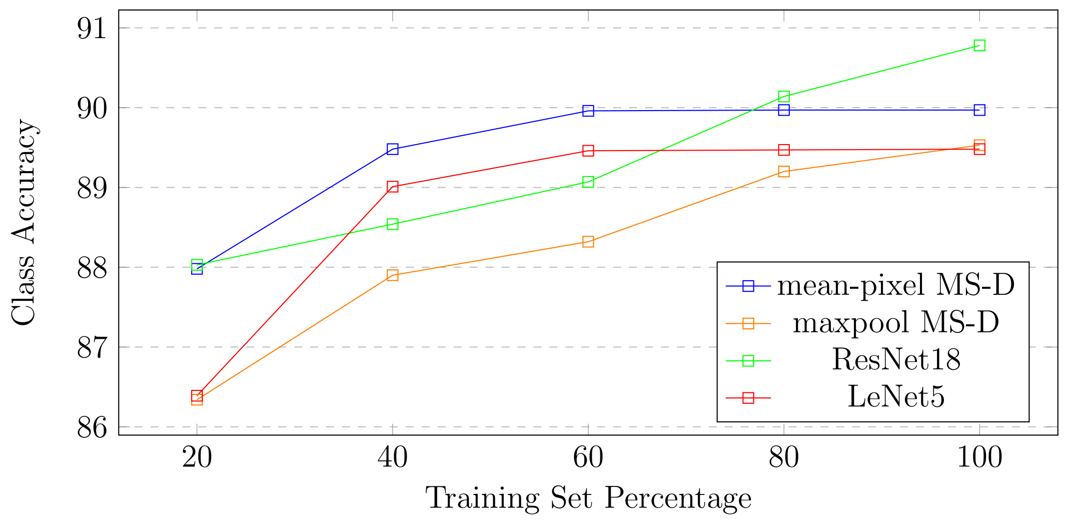

Figure 13: Graph showing the average class accuracy on the test set of different classification networks on the Fashion MNIST dataset, using different training set percentages. The figure is zoomed in on higher training set percentages in order to visualize relevant data.

Figure 14 shows the amount of used parameters for each network compared to their accuracy, when using the entire training set. Here it is shown that for more complicated datasets like Fashion MNIST, the use of the maxpool MS-D extension is also not beneficial. It has a class accuracy that is similar to that of LeNet5, while using more than a hundred times more parameters. The mean-pixel extension has a good performance, while using approximately 40 times less parameters compared to ResNet, and 6 times more parameters compared to LeNet5, which has a class accuracy that is much lower. This suggests that the mean-pixel MS-D extension has a good balance between the performance on class accuracy and the amount of used parameters.

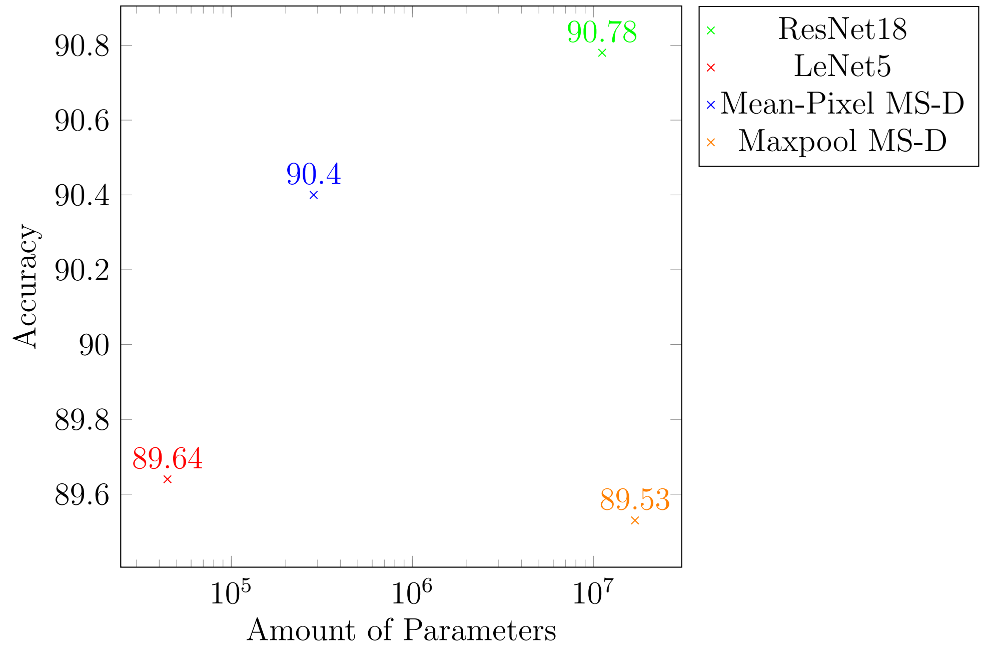

Figure 14: Scatter plot of the amount of parameters for each of the compared networks, compared to their class accuracy on the Fashion MNIST dataset

6 Discussion and Further Research

Extended MS-D networks for classification show promising results, since the class accuracy of the mean-pixel extension is similar to the ResNet18 network that was used in the experiments. On smaller training sets, this network performed better than all tested networks on the simple MNIST dataset. For the more complex Fashion MNIST dataset this is less the case, since the mean-pixel MS-D network only outperforms ResNet18 when using 40 and 60 percent of the training data.

In general, the mean-pixel extension performs similar compared to the more well known ResNet18 and LeNet5 networks. However, as of yet, there are some downsides to the usage of the extended MS-D networks. The largest issue is that the extended versions of MS-D are not very efficient. Training the networks takes a large amount of time compared to the other networks that were tested in this research. Optimization and fine-tuning of the code and the network could solve this issue, which would help in the research of other potential benefits of using MS-D for classification. The Maxpool extension of MS-D does not work well compared to the mean-pixel extension because the amount of parameters is too high. The fully connected layers are very large and have a large amount of parameters, which over-complicates the network. The advantage of the original MS-D for segmentation was that the network has a relatively small size, and that it is densely connected. Adding fully connected layers removes this advantage, which is clear when looking at the results in table 9 and 12. Other methods to improve the classification extension could be found. One possible method for extension of the MS-D network would be to add intermediate classifiers, like in the MSDNet which was mentioned in section 3. This could decrease training and execution time of the network. Since ResNet18 was able to outperform the mean-pixel MS-D extension for large amounts of data, it could be a good idea to explore how some parts of this architecture could be tranferred to the MS-D extension. ResNet18 uses skip connections, in order bypass layers that decrease the networks performance [Ban22]. Investigating whether this change in approach could benefit with performance of the extended MS-D network could be interesting.

Another subject that could be researched is the use of dilated convolutions. In this research the images were too small to make dilated convolutions useful, however it is possible to implement these convolutions in the extended MS-D versions. A dataset with larger and more detailed images would be a good fit for such a research. The dilated convolutions have proven to be very efficient for the original MS-D network for segmentation, so it would be interesting to see if the advantages on segmentation problems translate to classification problems.
Another issue during this research were the hardware constraints. Due to the size of the GPU storage, a network with a depth over 250 was impossible to test, and large networks of the extended MS-D versions have a very long execution time for depths over 100, due to forced small batch sizes. With better hardware or more data-efficient implementation of the network, networks with a higher depth could be explored, which could be very useful for datasets that are more complicated than the Fashion MNIST dataset.

7 Conclusions

In this thesis two methods for the extension of Mixed-Scale Dense (MS-D) CNNs for classification were explored. MS-D has been proven to be able to achieve good results with less parameters and layers compared to traditional networks, due to the use of dilated convolutions and dense connections. MS-D only existed for segmentation problems, where the result is an image, and not for classification problems.

First, research was done in order to find how the original MS-D network could be extended for solving classification problems. The first method for the extension uses a mean operation on the pixels of the image, with a softmax operation after the final layer of the network. The second extension uses a max-pooling operation, where the size of the image is reduced to the size of a single pixel, before passing through three fully connected layers, taking inspiration from the VGG16 network.

Secondly, it was researched whether the extended versions of the MS-D network for classification maintain the advantages of the original MS-D network, when applied to classification problems. Finally, the performance of the extended MS-D networks was compared to the performance of other well-known classification networks, by comparing their class accuracy and amount of used parameters. On a simple dataset, the mean-pixel MS-D extension does about as well as better-known classification networks, while using less parameters. When the training set is smaller and less data is available, it has slightly better results. This suggests that the advantages of using less parameters and dense connections that were found for the original MS-D network for segmentation slightly translate to classification problems. For the Maxpool MS-D extension this is not the case, since it performs the worst out of all tested networks. On a more complex dataset, the mean-pixel extension performs slightly worse, but shows similar results compared to the more well-known classification networks. The maxpool MS-D extension performs even worse on this dataset. The use of fully connected layers over-complicates the network, resulting in less accurate predictions and too many parameters.

The mean-pixel extension of MS-D for classification shows promising results for future research, since it performs comparably to well-known classification networks, while using less parameters. The network needs optimizing and the performance of other extensions could be investigated. Other datasets could be tested, in order to find whether dilated convolutions would contribute to an extended MS-D network for classification. For future research, it is noteworthy that the original benefits of the MS-D network should remain intact, since they have shown to have large impact on the performance of the network.

References

[Ala23] Richmond Alake. Loss functions in machine learning explained. *datacamp.com [Online]. Available: https://www.datacamp.com/tutorial/loss-function-in-machine-learning*, 2023.

[Bal22] Maciej Balawejder. Overfitting in deep learning. *Towards Data science [Online]. Available: https://towardsdatascience.com/overfitting-in-deep-learning-what-is-it-and-how-to-combat-it-9760d25ad05b*, 2022.

[Ban22] Siddhesh Bangar. Resnet architecture explained. *mediu,.com [online]. Available: https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d*, 2022.

[Bro21] Jason Brownlee. Gentle introduction to the adam optimization algorithm for deep learning. *Machine Leaning Mastery [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/*, 2021.

[DW20] Jinglong Du and Lulu Wang. Brain mri super-resolution using 3d dilated convolutional encoder–decoder network - scientific figure on researchgate. *researchgate.net [Online]. Available: https://www.researchgate.net/figure/The-illustration-of-the-2D-dilated-convolution-with-spatial-size-of-33-and-dilation*$_{f}ig1_{3}$38780378, 2020.

[KB15] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *conference paper at ICLR 2015*, 2015.

[LCB10] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[Nan23] Georgios Nanos. Neural networks: Pooling layers. *baeldung.com [Online]. Available: https://www.baeldung.com/cs/neural-networks-pooling-layers*, 2023.

[NH10] Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines vinod nair. volume 27, pages 807–814, 06 2010.

[NLZJ19] Zijia Niu, Wen Liu, Jingyi Zhao, and Guoqian Jiang. Deeplab-based spatial feature extraction for hyperspectral image classification. *IEEE Geoscience and Remote Sensing Letters*, 16(2):251–255, 2019.

[NMRW22] Rajendran Nirthika, Siyamalan Manivannan, Amirthalingam Ramanan, and Ruixuan Wang. Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study. *Neural Computing and Applications*, 34:1–27, 04 2022.

[PS17] Daniël M. Pelt and James A. Sethian. A mixed-scale dense convolutional neural network for image analysis. *www.pnas.org [Online]. Available: https://www.pnas.org/doi/full/10.1073/pnas.1715832114*, 2017.

[PyT23] PyTorch Team. torch. `https://pytorch.org/`, 2023. Version 2.0.1.

[PyT24a] PyTorch Contributors. torch.mean. `https://pytorch.org/docs/stable/generated/torch.mean.html`, 2024.

[PyT24b] PyTorch Contributors. torch.nn.CrossEntropyLoss. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, 2024.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arxiv.org [Online]. Available: https://arxiv.org/abs/1409.1556*, 2014.

[Tsa19] Sik-Ho Tsang. Review: Msdnet — multi-scale dense networks (image classification). *Towards Data Science [Online]. Available: https://towardsdatascience.com/review-msdnet-multi-scale-dense-networks-image-classification-4d949955f6d5*, 2019.

[Unz22] Diego Unzueta. Fully connected layer vs. convolutional layer: Explained. *builtin.com [Online]. Available: https://builtin.com/machine-learning/fully-connected-layer*, 2022.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arxiv.org [Online]. Available: https://arxiv.org/abs/1708.07747*, 2017.

[YK16] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arxiv.org [Online]. Available: https://arxiv.org/abs/1511.07122*, 2016.