DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Exploiting the Data Hierarchy with Geometry Aware Sparse Grids for Image Classification

Samuel Weber

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Exploiting the Data Hierarchy with Geometry Aware Sparse Grids for Image Classification

Ausnutzung der Daten-Hierarchie mit geometrisch bewussten dünnen Gittern für Bild-Klassifizierung

Author:	Samuel Weber
Supervisor:	Prof. Dr. Hans-Joachim Bungartz
Advisor:	Kilian Röhner, M.Sc.
Submission Date:	16.08.2019

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 16.08.2019 Samuel Weber

Abstract

Currently, the geometry aware sparse grids allow us to use only simple stencils for image classification on a normal resolution. In this thesis, I present the data hierarchy, a set of different coarsened images of the original data, that allow us to apply complex stencils on lower resolutions while still being able to process the original image through simple stencils. By exploiting the data hierarchy the accuracy in some cases is improved by nearly 5%.

I also present the class of hierarchical parent stencils that can work vertically on the data hierarchy and uses fewer interaction terms allowing us to establish a data hierarchy without necessarily increasing the number of grid points.

Smart use of the data hierarchy and multiple stencils allows us to apply sparse grid image classification in cases that would have been infeasible before.

Contents

1 Introduction

Today more and more people are using the internet and are owning devices with a growing number of sensors. More than $2.5 \cdot 10^{18}$ bytes of data are generated each day by all our devices and sensors [Mar18]. This makes classification techniques and machine learning a very important topic since the amount of data we produce is increasing even further each day. A huge amount of this data are plain images or images contained in videos. To compensate the data growth we need to improve our ways to classify them.

Geometry aware sparse grids were designed to solve image classification problems using sparse grids even though image classification is usually a high dimensional problem [Wae17]. Only grid points are added to the sparse grid, which are in specific data dimensions. A so-called stencil defines those dimensions. The stencil has a huge impact on the performance and accuracy of the resulting sparse grid.

In this thesis, I will present an attempt to improve the capability of geometry aware sparse grids by creating a data hierarchy of each image. The data hierarchy consists of coarsened images of each data entry. This allows us to apply complex stencils on lower resolutions, while still being able to use simpler stencils on the original image and thus reducing the number of grid points.

Additionally, the data hierarchy may also increase the accuracy. Through the coarsening of the image, some image features become more emphasized. For instance, for an image classification of a high-resolution image, it is not relevant whether a black pixel is right next to a more reddish one. Instead, it is rather more important whether a black area is next to a reddish area. Through the coarsening of the images, these areas become pixels, on which complex stencils can be applied, which would be infeasible on high resolutions. Therefore, the accuracy may be improved.

I will give a brief overview of geometry aware sparse grids and how classification problems are solved by it in Section 2. Then I will show firstly my approach of generating a data hierarchy and then defining a class of hierarchical parent stencils in Section 3. The implementation of the stencils and some changes to the SG++ Software to improve performance will be shown in Section 4. Finally, the performance and accuracy of the newly developed techniques are shown and analyzed on the MNIST and CIFAR-10 dataset in Section 5.

2 Fundamentals

In this chapter, I will explain the basic theory for a sparse grid image classification. First, the basic interpolation of a function using a full grid will be presented and then a transition to sparse grids will be made. Afterward, the transition from sparse grids to geometry aware sparse grids will be shown and then a possible way to solve classification problems using sparse grids will be described.

2.1 Full Grid Interpolation

Since a grid interpolation is a discretization technique, only a bounded function can be interpolated on arbitrary positions. Otherwise only a subdomain of the function is useable. Without loss of generality, the unit-hypercube $\Omega \in [0,1]^d$ is used as the domain for a d-dimensional function $f : \Omega \to \mathbb{R}$, which shall be interpolated. The space Ω can be discretized by a full grid of a resolution $n \in \mathbb{N}$, containing equidistant grid points x_i with a mesh size of $h = 2^{-n}$. An interpolant $u : \Omega \to \mathbb{R}$ can then be defined as a weighted sum of suitable d-linear basis function $\varphi_i : \Omega \to \mathbb{R}$ [Pfl10]:

$$f(\vec{x}) \approx u(\vec{x}) = \sum_i \alpha_i \varphi_i(\vec{x}) \tag{2.1}$$

with $\alpha_i \in \mathbb{R}$ being the weight for the basis function φ_i.

2.2 Hierarchical Basis Function

As can be seen in Equation 2.1, the performance of the interpolation is mainly dependent on a good choice of basis function. A basis is needed, such that as few as possible basis functions contain all necessary information so that basis functions that are not affecting the result strongly can be skipped and the basis functions should also be easily computable. Such a basis can be found by a hierarchical construction of the function space [Pfl10]. Starting from the one-dimensional case, the hierarchical basis will be derived and later on extended to support d-dimensional cases.

2.2.1 Hierarchical Basis Function in One Dimension

From the standard hat function [BG04]

$$\varphi(x) = \max\{1 - |x|, 0\} \tag{2.2}$$

we can derive a basis function through dilatation and shifting [Pfl10]:

$$\varphi_{l,i}(x) = \varphi(2^l x - i) \tag{2.3}$$

with the hierarchical level $l \in \mathbb{N}$ and an index $0 \leq i \leq 2^l$. With these modifications, it is achieved that the function is centered at the grid point $x_{l,i}$. This can be seen in Figure 2.1.

Now a set of piecewise linear functions V_n in which our interpolant u can be found is derived. V_n is the function space of a grid of level n. First, a set of indices is declared [Pfl10]

$$I_l = \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1 \land i \text{ odd}\}, \tag{2.4}$$

from which we can obtain the hierarchical subspace [Pfl10]

$$W_l = \text{span}\{\varphi_{l,i} : i \in I_l\}. \tag{2.5}$$

This subspace contains all basis functions, that is needed to extend V_{n-1} to V_n [BG04]. In other words, adding the hierarchical subspace W_l to a grid of level $n-1$ changes the level of the grid to n. With $V_1 = W_1$, we can sum up the subspace to obtain V_n:

$$V_n = \bigoplus_{l \leq n} W_l \tag{2.6}$$

A graphical representation of the complete process for the one-dimensional case can be seen in Figure 2.1.

The interpolant $u \in V_n$ can then be calculated with [Pfl10]

$$u(x) = \sum_{l \leq n, i \in I_l} a_{l,i} \varphi_{l,i}(x), \tag{2.7}$$

where the weight $\alpha_{l,i}$ is also being referred to as the (hierarchical) surplus.

2.2.2 Moving to the Multi-Dimensional Case

In the d-dimensional case, the basis functions are extended in a tensor product manner [Pfl10]

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) = \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j) \tag{2.8}$$

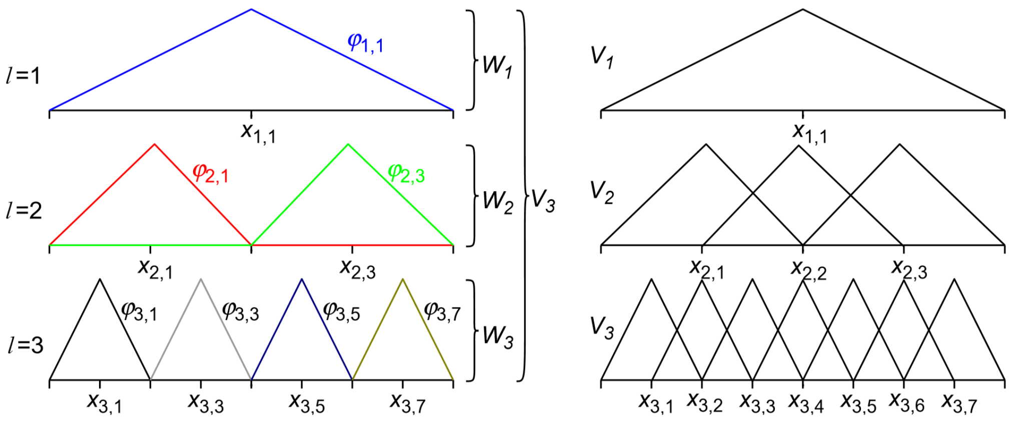

Figure 2.1: Construction of the piecewise linear function (right) space via hierarchical subspaces (left) in a one-dimensional case. Including all hierarchical subspaces up to a level l results in the function space V_n. Taken from [Pfl10].

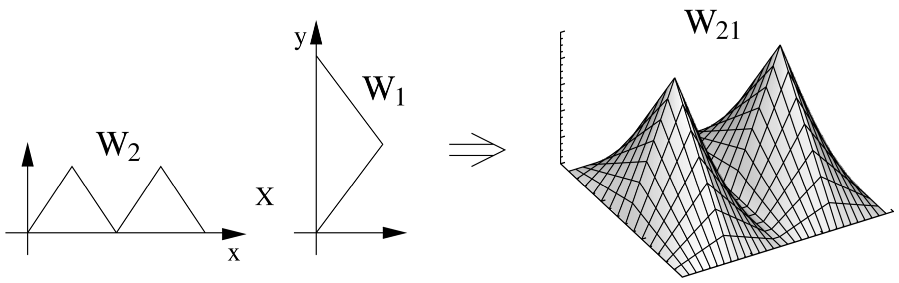

Figure 2.2: Tensor product of the subspaces in a two-dimensional case. Taken from [BG04].

where the vectors $\vec{l}$ and $\vec{i}$ contains the level and the index of each dimension. For the d-dimensional hierarchical subspace the index set must also be redefined to support multiple dimensions [Pfl10]:

$$I_{\vec{l}} = \{\vec{i} : 1 \leq i_j \leq 2^{l_j} - 1 \land i_j \text{ odd} \land 1 \leq j \leq d\}, \tag{2.9}$$

from which we can obtain the multi-dimensional subspaces [Pfl10]

$$W_{\vec{l}} = \operatorname{span}\{\varphi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\} \tag{2.10}$$

and finally the multi-dimensional function space of d-linear functions with a mesh size of h_n [Pfl10]

$$V_n = \bigoplus_{|\vec{l}|_\infty \leq n} W_{\vec{l}} \tag{2.11}$$

where $|\vec{l}|_\infty$ is the maximum norm defined as $|\vec{x}|_\infty = \max_j x_j$. For a two-dimensional case, a graphical representation of a hierarchical subspace is shown in Figure 2.2

The interpolant $u \in V_n$ can then be obtained with [Pfl10]

$$u(\vec{x}) = \sum_{|\vec{l}| \leq n, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}) \tag{2.12}$$

If the function f is smooth enough, the error of the full grid interpolation is [Pfl10]

$$||f(\vec{x}) - u(\vec{x})||_{L2} \in O(h_n^2) \tag{2.13}$$

but with a growth of basis functions in [Pfl10]

$$O(h_n^{-d}) = O(2^{nd}) \tag{2.14}$$

and thus enduring the curse of dimensionality, as the number of grid points grows exponentially with the number of dimensions.

2.3 From Full to Sparse Grids

Sparse grids are able to overcome the curse of dimensionality by including only certain hierarchical subspaces [BG04]. Only the hierarchical subspaces up to the diagonal of the desired level are included in a regular sparse grid, while a full grid would use a complete square [BG04]. The number of grid points that are not included is quite large, which can be seen in Figure 2.3.

The sparse grid space is, therefore, a slight alteration of V_n [Pfl10]

$$V_n^{(1)} = \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}, \tag{2.15}$$

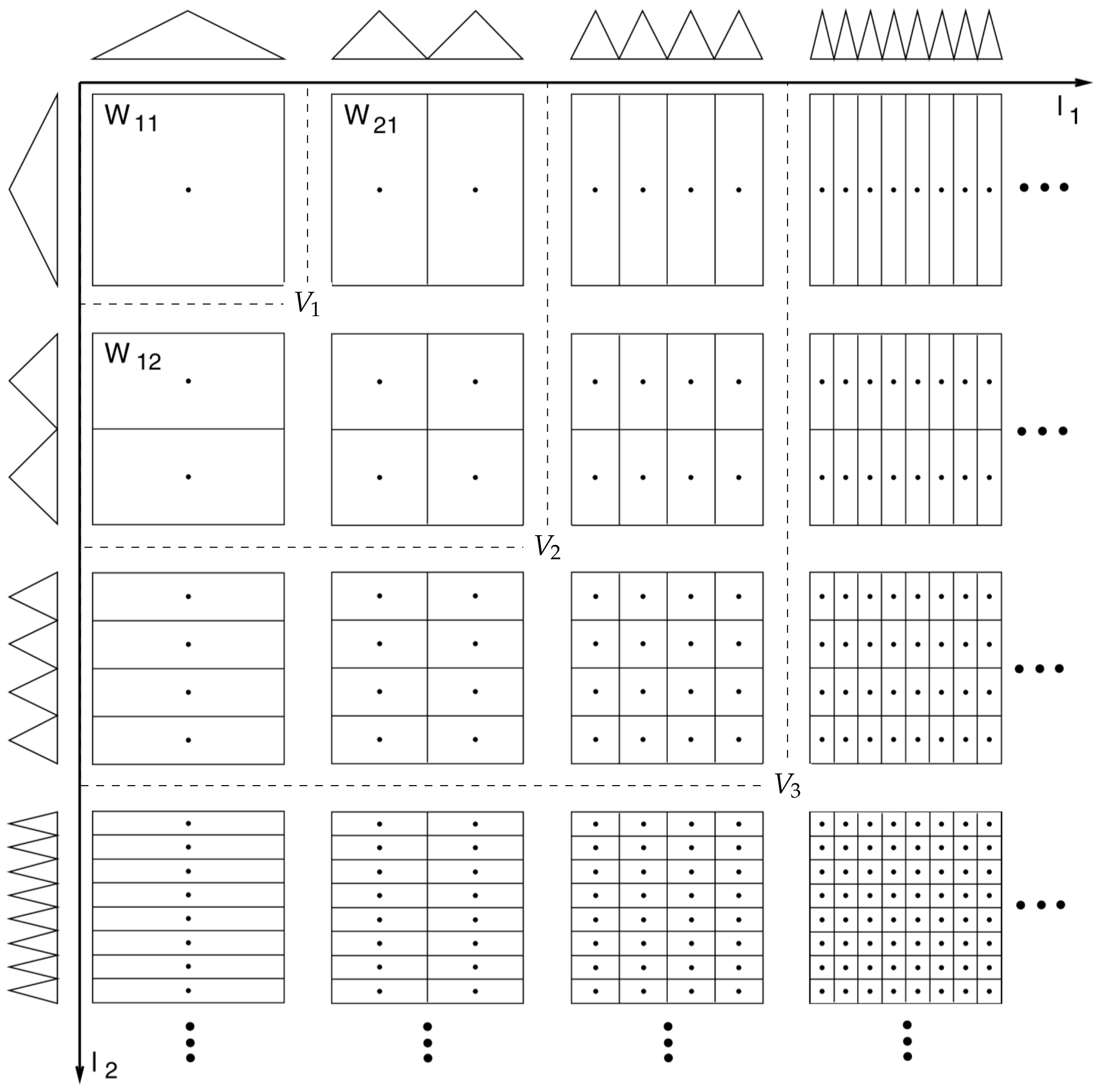

Figure 2.3: Grid combination scheme for two dimensions. The dots mark the position of the grid points in the corresponding hierarchical subspace. Image taken from [BG04] and modified to show the subspaces included in the function space V_n.

with $|\vec{l}|_1$ being the L1-norm defined as $|\vec{x}| = \sum_j^d |x_j|$. The interpolant $u(\vec{x}) \in V_n^{(1)}$ is then obtainable by [Pfl10]

$$u(\vec{x}) = \sum_{|\vec{l}|_1 \leq n+d-1, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}) \tag{2.16}$$

which leads to a significant reduction of complexity to $O(h^{-1}(\log h_n^{-1})^{d-1})$ basis function evaluations at the cost of slightly decreasing the accuracy to $O(h_n^2(\log h_n^{-1})^{d-1})$ [Pfl10].

2.4 Handling the Boundary

As can be seen in Figure 2.1 the current approach cannot handle boundary values of the function. To support boundary values boundary grid points could be added, but since this approach would lead to a huge increment of complexity, the basis function is slightly modified. The modified linear basis function is defined as [Pfl10]:

$$\varphi_{l,i}(x) = \begin{cases} 1, & \text{if } l = 1 \wedge i = 1 \\ \begin{cases} 2 - 2^l \cdot x, & \text{if } x \in [0, \frac{1}{2^{l-1}}] \\ 0, & \text{else} \end{cases} & \text{if } l \geq 1 \wedge i = 1 \\ \begin{cases} 2^l \cdot x + 1 - i, & \text{if } x \in [1 - \frac{1}{2^{l-1}}, 1] \\ 0, & \text{else} \end{cases} & \text{if } l \geq 1 \wedge i = 2^l - 1 \\ \varphi(x \cdot 2^l - i), & \text{else.} \end{cases} \tag{2.17}$$

Instead of having the value 0 at the boundary, we now have a default value of 1. As shown in Figure 2.4, through suitable scaling via the surplusses, boundary values can be considered.

2.5 Geometry Aware Sparse Grids

Although sparse grids can overcome the curse of dimensionality, a higher dimensionality will still increase the number of grid points. Thus, for high dimensional problems, such as image classification, a regular sparse grid is infeasible. For instance, a color image with a resolution of 8×8 corresponds to a space with $8 \cdot 8 \cdot 3 = 192$ dimensions. If a sparse grid of level three would be used, the resulting grid would have $74,497$ grid points and on level four it would have $9,659,649$ grid points. Therefore, already on small levels, the numbers of grid points is infeasible.

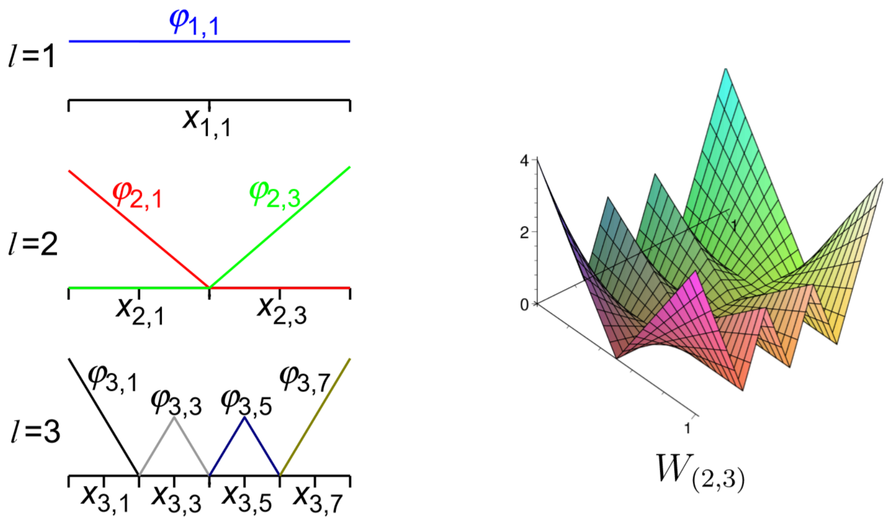

Figure 2.4: Treatment of the boundary with a modified linear basis function. Image taken from [Pfl10].

Waegemans introduced in his thesis [Wae17] an approach to reduce the number of grid points a lot without significant changes to the error rate by exploiting the following property of image data:

1. Image data contains information in the order of the dimensions.
2. The context (respectively the neighborhood) of a pixel contains most of the information.

The second property follows from the first, since dimensions must not be swapped, except the complete context of the pixel would be swapped too, for instance by applying rotations or scalings. Otherwise, the image would get unrecognizable even for humans.

Waegemans uses this geometric property, by forcing the sparse grid to only include hierarchical subspaces, that are adding grid points between dimensions belonging to each others neighborhood. Of course, this includes hierarchical subspaces, that are only adding grid points to one dimension. Hierarchical subspaces are adding grid points between all dimensions, that have a level of two or higher in the level vector (cf. Figure 2.3) [Wae17]. In all other hierarchical subspaces, those who only have one level higher than or equal to two, only the resolution of one dimension is improved. For instance, in a three-dimensional case, the hierarchical subspace $W_{1,3,2}$ adds grid points between the x_2 and x_3 axis, while the subspace $W_{1,1,3}$ only increases the resolution on the x_3 axis. Hence, if it is known that only the second and the first dimension correlate, the

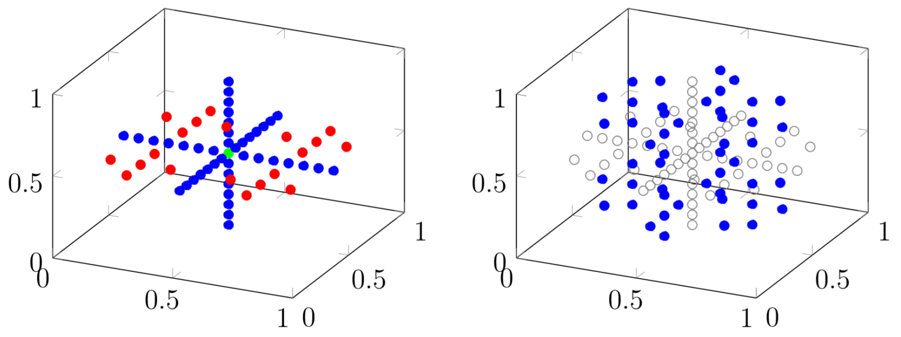

Figure 2.5: On the left, the grid points of a geometry aware sparse of level four with the interaction terms $T = \{\emptyset, \{x\}, \{y\}, \{z\}, \{x, y\}\}$ are shown. On the right, the grid points are shown, that are not included due to the interaction terms. Image taken from [Wae17].

subspace $W_{1,3,2}$ can be left out, probably without affecting the error rate much. For a similar case, the resulting grid points and also the not included grid points are shown in Figure 2.5.

2.5.1 Mathematical definition of sparse grids

To describe which dimensions belong to each other's context interaction terms $t \subseteq D$ are used, with $D = \{d_1, d_2, ..., d_n\}$ being the set of dimensions [Kre16]. The set of interaction terms is, therefore, a subset of the power set of dimensions:

$$T \subseteq 2^D \tag{2.18}$$

Let $\zeta : \mathbb{N}^d \to 2^D$ be a function that returns the interaction term modeled by the hierarchical subspace $W_{\vec{l}}$ as discussed above, defined as [Wae17]:

$$\zeta(\vec{l}) = \{d_i : l_i > 2 \land i \in [d]\}. \tag{2.19}$$

With this, the interaction-based sparse grid function space [Wae17]

$$V^T = \bigoplus_{W_{\vec{l}} \in V^{(1)} \land \zeta(\vec{l}) \in T} W_l \tag{2.20}$$

can be obtained.

To calculate the total number of grid points, that the corresponding geometry aware sparse grid based on the interaction set has, the following equation may be used

k	Level 2	Level 3	Level 4	Level 5
0	1	1	1	1
1	2	6	14	30
2	0	4	20	68
3	0	0	8	56
4	0	0	0	16
5	0	0	0	0

Table 2.1: Some values of g_k for different levels. g_k returns the number of hierarchical subspaces, which corresponds to an interaction term of size k. Observe, that subspaces that are modeling an interaction term of size k are only included in levels greater than or equal to $k+1$.

[Wae17]:

$$n = \sum_{i=0}^{max\{|t|:t\in T\}} |S_i| \cdot g_i \tag{2.21}$$

where S_i is the set of interaction terms with a cardinality of i

$$S_i = \{t : t \in T \wedge |t| = i\} \tag{2.22}$$

and $g_k : \mathbb{N} \to \mathbb{N}$ is a function that returns the number of grid points in a hierarchical subspace, that models an interaction term of size k for a grid level m [Wae17]:

$$g_k(m) = \begin{cases} \sum_{i=k+1}^{m} 2^{i-1} \cdot \binom{i-2}{i-k-1}, & \text{if } k \geq 1 \\ 1, & \text{if } k = 0 \end{cases} \tag{2.23}$$

For a comprehensive explanation of this equation, see [Wae17]. A list of values for g_k for the sparse grid level two, three and four is given in Table 2.1. From the values of g_k, one can see that an interaction term of size k is only included in a sparse grid of level $k+1$.

2.5.2 Stencils

The rule through which the interaction set is generated is referred to as stencil [Wae17]. Waegemans, introduced the direct neighbor stencil (DN-stencil), which adds one interaction term each, for all pixels with a hamming distance of one [Wae17]. Hence, the direct neighbors are the pixels to the left, right, top and down. Therefore there are $(x-1)y + x(y-1)$ interaction terms of size two, which also is the largest size of interaction terms.

	DN-stencil	ColDN-stencil
$\lvert S_1 \rvert$	xy	$3xy$
$\lvert S_2 \rvert$	$(x-1)y + x(y-1)$	$3(x-1)y + 3x(y-1) + \binom{3}{2}xy$
$\lvert S_3 \rvert$	0	xy
$\lvert S_4 \rvert$	0	0

Table 2.2: Sizes of the interaction set partitioned by the size of the interaction terms. x and y are the width and the height of the image. For a comprehensive derivation of these sizes for the stencils see [Wae17].

If the image is colored, the colored direct neighbor stencil (ColDN-stencil) includes interaction terms for each direct neighbor of the same color. Interactions between different colors are only included for the same pixel. For an RGB-colored image, this leads to a maximum interaction size of three. Since all child hierarchical subspaces must also be included in the sparse grid the power set of $\{r_i, g_i, b_i\}$ must be a subset of the interaction terms, where r_i is the index of the red color dimension of the pixel i, g_i and b_i analogously. Therefore, for each pixel i the set of interaction $\{\emptyset, \{r_i\}, \{g_i\}, \{b_i\}, \{r_i, g_i\}, \{r_i, b_i\}, \{g_i, b_i\}, \{r_i, g_i, b_i\}\}$ are added to the interaction set T. The resulting number of interaction terms together with the ones from the DN-stencil are shown in Table 2.2.

Waegemans also proposed the more complex cube stencil [Wae17]. It contains all possible interactions terms of pixels in a square of a specific size. Thus it acts like a regular sparse grid, which is applied only on the sub-image. The number of interaction terms of this stencil grows exponentially with the square size, as all possible interaction terms of the pixels inside the cube have to be included. This is equivalent to the power set of the pixels. I use the notation 'cube-l' to denote a cube stencil of length l.

2.6 Spatial Adaptivity

Only for functions that are smooth enough, the sparse grid scheme defines a good set of grid points [Pfl10]. But for functions that have strong local fluctuations, the sparse grid interpolation could be erroneous. To compensate the error a higher level of the sparse grid may be used, but this would increase the number of grid points enormously. Instead, it would be preferable to only add grid points where the local fluctuations are.

This can be done by a refinement of a grid point that covers at least some parts of the critical area. The refinement adds all $2 \cdot d$ children of a grid point. The children are all the grid points of the next sparse grid level, which basis functions cover the same area that the parent grid point has covered. To add those grid points it must be ensured,

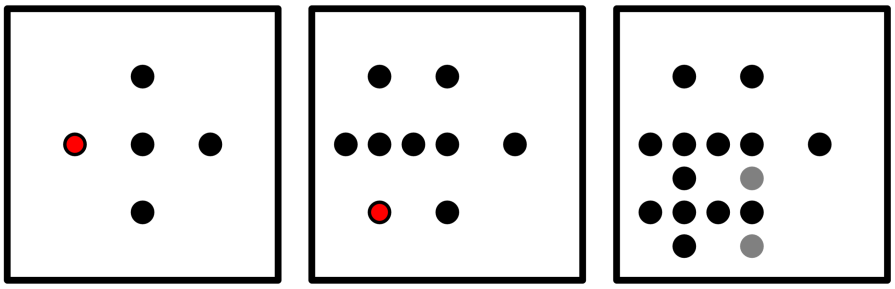

Figure 2.6: Refinement of a grid point (marked in red) by adding all of its $2 \cdot d$ children and if necessary adding the non-existent parent grid points (marked in gray) for a sparse grid of level two.

that all hierarchical ancestors exist [Pfl10]. Therefore, in some cases not only children of the grid point that is getting refined need to be added.

In comparison to the sparse grid scheme, the refinement grid points are chosen a posteriori, hence after the sparse grid is generated. Critical areas may be detected through the values of the surplusses or via other techniques. For more details of different refinement criterions see [Pfl10].

2.7 Classification using Sparse Grids

A classification problem can be solved with sparse grids by applying one sparse grid density estimation per class.

2.7.1 Density Estimation using Sparse Grids

The data of one class is interpreted as random samples from a probability distribution which density function must be approximated. Areas with a lot of samples will, therefore, have a large density while areas with small or none samples will have a small density. Hence, the value of the interpolant u, obtained through the sparse grid density estimation of class i, at a point x is the probability that the data represented through x is of class i.

The sparse grid density estimation is formulated by the equation [Peh13]

$$(R + \lambda C)\vec{\alpha} = \vec{b} \tag{2.24}$$

with R being a matrix of L2 scalar products of the basis functions with $R_{i,j} = \langle \varphi_i, \varphi_j \rangle_{L2}$. The vector $\vec{\alpha}$ contains the surplusses of the basis functions and b_i is the average value of the basis function φ_i over all the data in the training set of the particular class. $\lambda \in \mathbb{R}$

is a scalar to control how much the regularization matrix $C_{i,j} = \langle \Lambda\varphi_i, \Lambda\varphi_j \rangle$ affects the result. Commonly, Λ is chosen, such that the matrix C is equal to the identity matrix I. Therefore, no complex calculation is needed and a smoother function is advocated (c.f [Pfl10]). [Peh13]

This equation for the density estimation has the benefit, that the matrix C is of size $n \times n$ with n being the number of grid points. Therefore, the complete system of linear equations is solvable in $\mathcal{O}(n^3)$ and thus the system's complexity is independent of the size m of the particular class training set [Peh13]. Only for the calculation of the vector $\vec{b}$ the training set size is relevant. To calculate the average we need to iterate over each item once and this for all grid points. Thus the calculation of the vector needs $\mathcal{O}(n \cdot m)$ steps. Hence, the total complexity for the system of linear equations is in $\mathcal{O}(n^3 + n \cdot m)$.

An improvement of the complexity can be achieved by decomposing the matrix into an online and an offline system. Through the decomposition, the system can be solved in $\mathcal{O}(n^2)$ steps. This is a huge improvement especially if the decomposition can still be used after a refinement step, which is, for instance, the case when a Cholesky decomposition was used.

2.7.2 Classification using Density Estimation

The result of the classification can then be obtained by evaluating each approximated density estimation u_i for each class $i \in \mathbb{N}$ at the point x. The result is then gained in a "one-hot" manner, meaning that the resulting label is the class which density estimation was the highest, no matter how close the other values were [Pfl10].

$$\text{label}(x) = \arg\max_i u_i(x) \tag{2.25}$$

The density estimation can also be used to express how certain the result is. If, for instance, the density of only one class is high, the result is likely to be accurate. On the other hand, if all density estimations return a similar value, the result is very likely to be wrong.

3 Exploiting the Data Hierarchy

The basic idea in this thesis is to transform each data point so that it becomes a data hierarchy consisting of coarsened versions of the original data. With this data hierarchy complex stencils that are infeasible on high resolutions may be applied to the coarsened versions of the data, while less complex stencils can still be applied on the full resolution.

I will propose a general approach to create a data hierarchy and explain why it can increase the accuracy and also increases the performance of the classification, in the first part of this chapter. Then in the second part, I will define the class of hierarchical parent stencils, that may be used for interactions between the different layers of the data hierarchy.

3.1 Data Hierarchy

The transformation of the images into a data hierarchy is done by appending coarsened versions to the original data point. I define the data hierarchy as the set of layers $L = \{l_1, l_2, \dots, l_n\}$. I also assume that the data hierarchy is sorted in descending order. Thus the layer l_i will have a greater resolution and hence a greater dimensionality than the layer l_{i+1}.

The layers' resolutions are stored in the matrix $A \in \mathbb{N}^{n \times \sigma}$ with σ being the total numbers of axes of the data. Therefore, if the data is a colored image ($\sigma = 3$), then $a_{2,1}$ will be the length, $a_{2,2}$ will be the height and $a_{2,3}$ will be the number of color channels of the second layer.

Using a data hierarchy to decrease the number of grid points might seem strange at first since the dimensionality of our classification problem is increased and the advantage of using more complex stencils on layers with lower resolution can also be accomplished by working on a coarsened image directly and not generating the whole data hierarchy. But if the image has also some features that are only recognizable on high resolutions, e.g. a one-pixel thin line, then working on the coarsened image is not applicable as an important feature of the image may be lost. Therefore the data hierarchy can be exploited to work on several resolutions of the image with different stencils. A complex stencil can be applied on the coarsened image while a simple stencil could be used to detect features of the high-resolution image.

The data hierarchy may also be used to improve accuracy. If a feature can only be easily detected on a lower resolution, but this one feature would not be enough to do a correct classification, the feature can not be used without a data hierarchy to improve accuracy. Such a feature would be, for instance, the average color of an image. On a layer of resolution 1×1, this feature would be clearly visible while most classification problems can not be correctly solved only based on the average color. Through the data hierarchy though, we can include the 1×1 version as well as the original image. Hence, this feature of the image can be used for the classification along with the other features of the original image so that the accuracy is improved.

3.1.1 Generating the Data Hierarchy

A good data hierarchy does not include unnecessarily layers, as they will only increase the dimensionality. On the other hand, it should include layers which are necessary to either emphasize a feature or to enable the use of a complex stencil without adding too many grid points. Also, all layers should be coarsened as strong as possible, without losing the desired features, to reduce the number of added dimensions.

If no special domain knowledge is available to decide which resolution to pick for the layers, a generally applicable approach could be to half each axis of the original data from layer to layer until the last layer has a size of one in each axis. To guarantee that the last image is of this size, the numbers are always rounded up.

Of course, it is normally desired to exclude the color information axis in this approach. For the reason of simplicity, I will assume, without loss of generality, that the data has no color information axis.

The approach can be expressed by the recursive equation:

$$l_{i+1} = \text{coarsen}_{\lceil \vec{a}_i \cdot \frac{1}{2} \rceil}(l_i)$$

where $coarsen_{\vec{s}}(x)$ is a function that coarsens the data x to the resolution $\vec{s}$. This leads to a data hierarchy of size $|L| = \max\{\lceil \log |\vec{a}_i|_\infty \rceil\}$ and a dimensionality of

$$d = \sum_{i=1}^{|L|} \prod_{k=1}^{\sigma} a_{i,k} = \sum_{i=1}^{|L|} \prod_{k=1}^{\sigma} \lceil \frac{a_{1,k}}{2^i} \rceil. \tag{3.1}$$

If the data consists of plain images and the resolutions are of a power of two, the equation can be written in its closed form:

$$d = \frac{a_{1,1} \cdot a_{1,2}}{3} \cdot (1 - 4^{-|L|}) \tag{3.2}$$

To decrease the number of added dimensions by this approach it is recommended to check the generated dataset and decide whether all of the layers are really necessary.

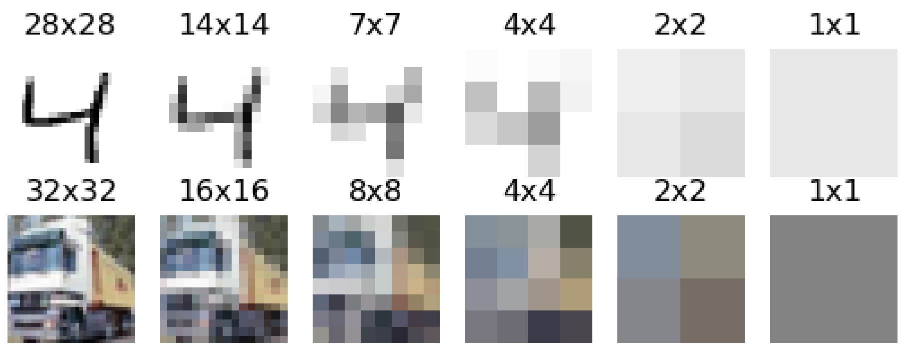

Figure 3.1: Data Hierarchy examples generated through the proposed general approach. The first row shows a sample from the MNIST dataset and the second row from the CIFAR-10 dataset. Both datasets are introduced in detail in Section 5.

If two layers are nearly identical or do not include any important features, the layer with the highest resolution should be left out, as it is affecting the number of added dimensions the most. For instance, each data point of a 28×28 grayscaled MNIST-image would have $1,050$ dimensions and in the case of a 32×32 colored CIFAR-10-image $4,095$ dimensions. As it can be seen on the resulting data hierarchies in Figure 3.1, in the case of the MNIST-sample the layer of size 14×14 has nearly no difference to the original image. Since it does not reveal any important feature of the image, one of the layers should be excluded. As the original image has the highest resolution, it is excluded and thus the dimensionality is reduced to 266.

3.2 Hierarchical-Parent Stencil

Until now, only interactions working horizontally on each layer of the data hierarchy have been considered. In this subsection, the class of hierarchical parent (HP) stencils is introduced and analyzed. This stencil type is working vertically on the data hierarchy and thus is putting the different layers in relation to each other.

The basic idea behind this stencil type is to use the fact, that the neighborhood of a pixel is completely contained by another pixel on a more coarsened layer. Therefore, the HP-stencils follow the same intuition as the DN-stencil, with the difference, that in the DN-stencil the direct neighbors are considered explicitly while the HP-stencils are considering them implicitly. Through this implicit interaction, the HP-stencils use fewer interaction terms than the DN-stencil and has also the ability to include larger neighborhoods without needing larger interaction terms as for instance the cube stencil uses. Of course, an implicit interaction will not be as accurate as an explicit one, but through this, the number of grid points may be reduced, without affecting the error rate strongly.

The HP-stencils have in common that only interactions between child and parent dimensions may be included. The parent-child relation is defined as:

$$R = \left\{ (p_{i,\vec{x}}, p_{j,\vec{y}}) : i < j \wedge \left(\frac{x_k}{a_{i,k}}, \frac{x_k + 1}{a_{i,k}} \right) \cap \left(\frac{y_k}{a_{j,k}}, \frac{y_k + 1}{a_{j,k}} \right) \neq \emptyset \right\} \tag{3.3}$$

where $p_{i,\vec{x}}$ is the null-based index of the pixel on the layer l_i that is positioned at the point $\vec{x} \in \mathbb{N}_0^\sigma$ with $0 \leq x_k < a_{i,k}$. Hence, two pixels are in the parent-child relation only if the coarsened pixel overlaps with the area of the pixel on the layer with the higher resolution. If the child layer's resolution is not dividable by the resolution of the parent, a pixel may have multiple parents, since multiple pixels overlap its area.

3.2.1 All-Hierarchical-Parent

Now to look at a concrete stencil, we define the all-hierarchical-parent (AHP) stencil. It is called so because it contains all possible hierarchical-parent interaction terms up to a cardinality of two. This means, that the set of interaction terms is

$$T = \{\emptyset\} \cup \{\{p_{i,\vec{x}}\} : i \leq |I| \wedge x_k \leq a_{i,k}\} \cup \{\{e, f\} : eRf\}. \tag{3.4}$$

Thus we get the following numbers of interaction terms (again partitioned by the interaction terms cardinality):

$$\begin{aligned} |S_0| &= 1 \\ |S_1| &= d \\ |S_2| &= \sum_{i=0}^{|L|} \prod_{j=0}^{\sigma} a_{i,j} \cdot (|L| - i) \end{aligned}$$

In the case of a grayscaled 32×32 image and a grid level of three, this would lead to $33,679$ grid points. These are too many grid points and thus the stencil will be infeasible on high resolutions. Especially the number of interaction terms of size two is the problem. Through the term $|L| - i$ the image with the highest resolution affects the result the most.

3.2.2 Next-Hierarchical-Parent

To reduce the number of grid points, the stencil can be modified, to only include interaction terms between certain layers, instead of connecting each layer to all other

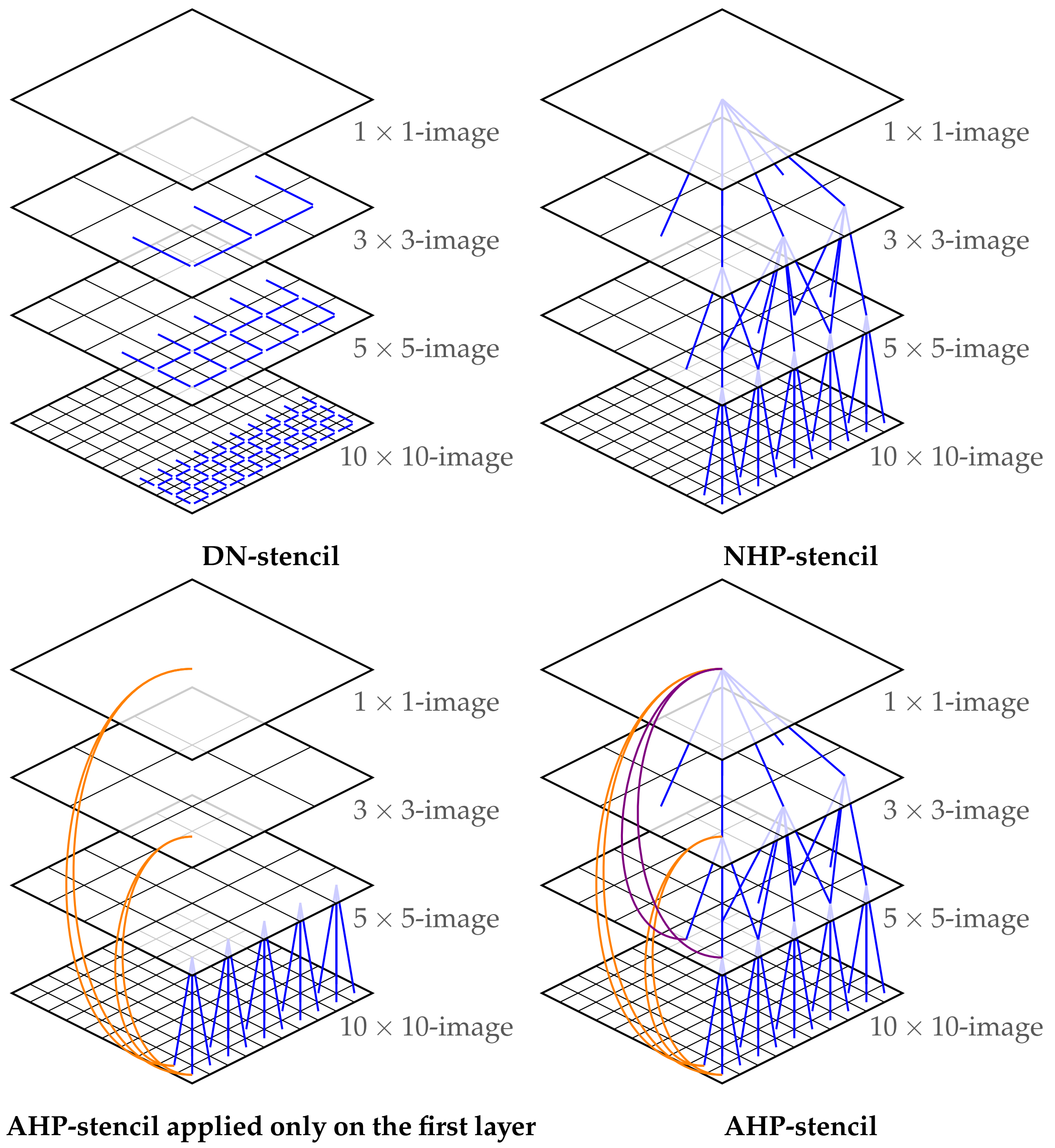

Figure 3.2: Schematics of the different stencils on a data hierarchy. Not all interaction terms are displayed for the reason of simplicity. Observe, that some pixels have multiple parents in the layer with resolution $\{5 \times 5\}$.

layers. A possible rule would be to only include the parents of the next layer. The resulting stencil will be referred to as next-hierarchical-parent (NHP) stencil and will have the set of interactions

$$\begin{aligned} T =&\{\emptyset\} \\ &\cup \{\{p_{i,\vec{x}}\} : i \leq |I| \wedge x_k \leq a_{i,k}\} \\ &\cup \{\{e, f\} : eRf \wedge e \in l_j \wedge b \in l_{j+1} \wedge j \leq |I| - 1\}. \end{aligned}$$

With the new set of interaction terms, the NHP-stencil can, therefore, be partitioned into

$$\begin{aligned} |S_0| &= 1 \\ |S_1| &= d \\ |S_2| &= \sum_{i=0}^{|L|} \prod_{j=0}^{\sigma} a_{i,j}. \end{aligned}$$

The critical term is gone and thus the number of grid points on a 32×32 grayscaled image is reduced to $13,647$. This is especially good since the DN-stencil applied on every layer in the data hierarchy would have $18,606$ grid points. This is achieved through the implicit comparison by the HP-stencils.

The smallest amount of grid points is generated if the layers' resolutions are of a power of two. In this case, each pixel has only one parent and thus at most two interaction terms of size two attached to it. Thus, the number of interaction terms of size two is smaller than for a DN-stencil, where each pixel has up to four interaction terms of size two. If the proposed general approach is used to generate the data hierarchy, then the number of dimensions/pixels in the data hierarchy will be lower than twice the dimensions of the original image without data hierarchy (c.f. Equation 3.2).

On the other hand, a DN-stencil without data hierarchy can have fewer grid points, as the number of interaction terms of size one is lower than with a NHP-stencil and a data hierarchy. Thus, for a low-level sparse grid, the number of grid points can be higher if a NHP-stencil with data hierarchy is used. But with higher sparse grid levels the number of interaction terms of size one becomes less important than the interaction terms of size two, since the number of hierarchical subspaces added to the grid using interaction terms of size two grows faster than those of size one, as it can be seen in Table 2.1. The number of grid points for different sparse grid levels is shown in Figure 3.3 and for different resolutions in Figure 3.4. As it can be seen in them, adding the data hierarchy and applying the NHP-stencil does not increase the number of grid points significantly for levels greater than or equal to three. This makes the NHP-stencil a good candidate if a data hierarchy should be established to apply a complex stencil on

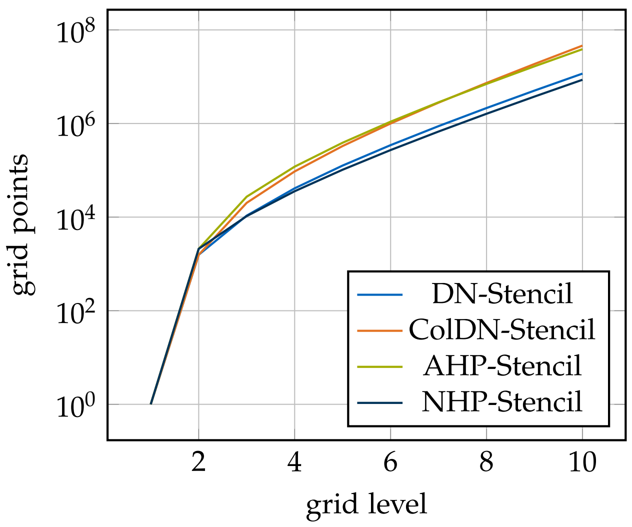

Figure 3.3: This diagram shows the number of grid points of a geometry aware sparse grid on a 28×28 image in correlation with the sparse grid level. The number of grid points for the DN- and ColDN-stencil is without a data hierarchy, while the number for the HP-stencils is with a data hierarchy generated through the proposed approach.

a lower resolution because the number of interaction terms of size one does not grow with an additional stencil. Therefore, if a combination of stencils is used on different layers, it is better to combine them with a NHP-stencil than with a DN-stencil. Keep in mind that the NHP-stencil in the diagram uses the general data hierarchy approach, hence a custom data hierarchy may reduce the number of grid points even further.

Another way to reduce the number of grid points the AHP-stencil generates is to exploit the data hierarchy by applying it only on some layers. We can, for instance, apply the NHP-stencil on the lowest layer and then the AHP-stencil on all other layers. Thus, reaching a similar structure, except that the image with the highest resolution which affects the number of grid points the most is not multiplied by the number of layers. Hence, the amount of grid points is reduced.

3.2.3 Handling of Color Information

Color information can be handled analogously as by the DN-stencil. The different color images are handled in the specific stencil manner, while interactions between different colors are only included in each pixel itself. The changes to the number of interaction terms are identical as in the case of the transition from the DN to the ColDN-stencil

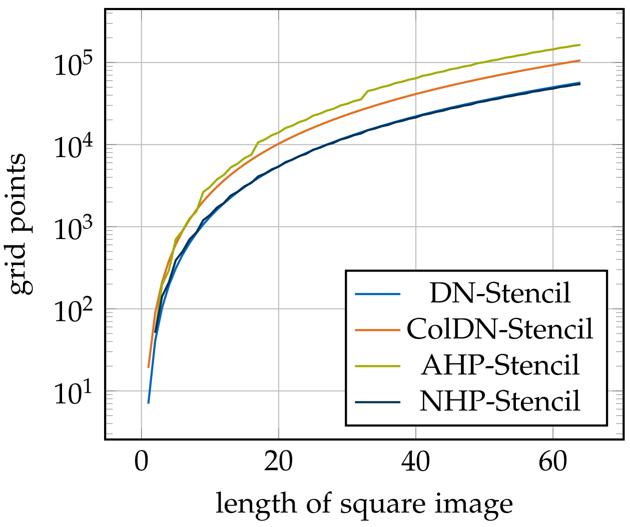

Figure 3.4: In this diagram, the number of grid points in a level three geometry aware sparse grid in correlation with the image resolution is shown. As in Figure 3.3, the DN-stencil and ColDN-stencil are applied to the original image and are not using a data hierarchy while the HP-stencils are using a data hierarchy generated through the proposed approach. Observe the large increment of grid points when a new layer is added in the case of the AHP-stencil.

4 Implementation

In this thesis, I used the SG++ library [PPB10], which delivers all necessary methods for an image classification using geometry aware sparse grids. First, I will show how the dataset has been modified. Then I will present the algorithm developed for the class of hierarchical stencils and finally, I will explain some modifications I made to the SG++ pipeline.

4.1 Data Preprocessing

All of the data preprocessing was made in Python using NumPy [vCV11], OpenCV [Bra00] and Keras [Cho+15]. NumPy delivers a helpful n-dimensional array, which Keras and OpenCV are using. OpenCV is a library for computer vision delivering many methods for image manipulations, whereas Keras is a high-level library for machine learning, which I used among other things to apply a data augmentation (c.f. Section 5.1).

4.1.1 Generating the Data Hierarchy

The datasets were loaded through Keras, all data were contained in NumPy arrays. The data has been resized according to the proposed approach for generating a data hierarchy. Each coarsened image has been generated through the OpenCV 'resize' method. The images were then concatenated sorted by their resolutions. Lastly, the data has been normalized to range between zero and one and then saved in an ARFF-format, so that it could be processed by the SG++ data-driven pipeline.

4.2 Changes to the SG++ Data-Driven Pipeline

The geometry aware sparse grid configuration was extended to support multiple dimensions so that an arbitrary data hierarchy could be used by the stencils. To exploit the data hierarchy a finer stencil control was needed so that multiple stencils could be defined and applied on different layers. Therefore, the stencil definition has been changed into an array of stencils, where each stencil object had a stencil type and a set of indexes corresponding to the layers in the data hierarchy attached to it. An example configuration file is shown in Listing 8.1.

The grid factory, which generated all interactions based on the stencil configuration has been, then, changed to combine all resulting interaction terms of each stencil together. To enforce that each interaction term can only be added once, the data structure of the interactions has been changed to a set. This was necessary, as duplicated interaction terms would lead to wrong results in Waegemans algorithm for the evaluation of geometry aware sparse grids [Wae17].

To speed the calculation up, the evaluation of the SG++ classifier has been changed to a matrix-based processing and to use the interaction-based evaluation method. Previously each data entry was evaluated individually. The performance gain was significant, especially on larger grids, as the interaction-based evaluation works at a faster order [Wae17]. In some cases, the evaluation was 20 times faster than before.

4.3 Hierarchical Parent Stencil

All previously developed stencils have been redesigned to work on an arbitrary number of axes in the data. Thus theoretically video data could be processed by them as well. Therefore, the HP-stencils should also be able to work on arbitrary many axes of the data.

The algorithm for the HP-stencils uses a recursive helper function, to find a parent in a different layer that overlaps with the same area. The parent is calculated by modifying the parent-child relation (c.f. Equation 3.3). The sets are multiplicated with a_j, k, so that the relation becomes

$$R = \left\{ (p_{i,\vec{x}}, p_{j,\vec{y}}) : i < j \wedge \left(\frac{x_k \cdot a_{j,k}}{a_{i,k}}, \frac{(x_k + 1) \cdot a_{j,k}}{a_{i,k}} \right) \cap (y_k, y_k + 1) \neq \emptyset \right\}. \qquad (4.1)$$

Now this relation can be used to calculate the parents' positions directly. There are two possible cases per axis k of the data:

1. The set $\left(\frac{x_k \cdot a_{j,k}}{a_{i,k}}, \frac{(x_k+1) \cdot a_{j,k}}{a_{i,k}} \right)$ does not contain a whole number.
2. The set $\left(\frac{x_k \cdot a_{j,k}}{a_{i,k}}, \frac{(x_k+1) \cdot a_{j,k}}{a_{i,k}} \right)$ does contain a whole number.

In the first case, there exists only one parent in this axis of the child and the position in the axis can be calculated by rounding the minimum or maximum of the set down. In the second case, there exist two parents in this axis and their position in the axis can be calculated by rounding both the minimum and the maximum of the set down. Since a set does not contain duplicated entries and in both cases, the positions are calculated by rounding the terms down, we do not need to check in which case we are. The algorithms are presented in Algorithm 1 and Algorithm 2.

Algorithm 1 HierarchicalParent

```
interactions $\leftarrow \emptyset$
**for** $i = 1$ to $|L|$ **do**
    **if** stencil should be applied on layer $l_i$ **then**
        **for** $j = i + 1$ to $|L|$ **do**
            **for** all coordinates $\vec{x}$ in layer $l_i$ **do**
                GetParentAndAddToInteractions($\vec{x}$, $i$, $j$, 1)
            **end for**
        **end for**
    **end if**
    **if** stencil is NHP-stencil **then**
        **return** interactions
    **end if**
**end for**
**return** interactions
```

Algorithm 2 The function GetParentAndAddToInteractions($\vec{x}$, i, j, k) is used to find all parents on the layer l_j of a child at position $\vec{x}$ and layer l_i. This function works recursively over all axes of the data.

```
**if** $k \leq \sigma$ **then**
    ratio $\leftarrow \frac{a_{j,k}}{a_{i,k}}$
    $\vec{y}_k \leftarrow \lfloor \vec{x}_k \cdot \text{ratio} \rfloor$
    GetParentAndAddToInteractions($\vec{x}$, $i$, $j$, $k + 1$)
    $\vec{y}_k \leftarrow \lfloor () \vec{x}_k + 1) \cdot \text{ratio} \rfloor$
    GetParentAndAddToInteractions($\vec{x}$, $i$, $j$, $k + 1$)
**else**
    interactions $\leftarrow$ interactions $\cup \{(p_{j,\vec{x}}, p_{k,\vec{y}})\}$
**end if**
```

5 Classification Results

In this chapter, I will analyze the accuracy gains of the data hierarchy and the hierarchical parent stencils, by applying them to some real-world datasets. First, I will show some of the modifications that I applied to the datasets. Then I will start presenting the results on the MNIST, fashion-MNIST and CIFAR-10 dataset.

5.1 Data Augmentation

A common problem in data mining is overfitting. Overfitting can occur when the dataset does not provide enough data points (samples). The problem of overfitting becomes apparent when a classification technique has a good accuracy on the training examples, but when applied on the testing samples, the accuracy is low. In such cases, the classification technique does not learn the desired task, instead, it uses the available variables it should set to memorize the training data. Thus a good rule of thumb is to have ten times more data points than variables.

In a sparse grid density estimation the number of grid points corresponds to the number of variables that must be set (cf. Equation 2.24). Since for each class, one density estimation must be applied and the number of grid points, as it can be seen in Figure 3.3, is quite high, our training sets must also be huge. As many datasets are not containing enough samples, new data points must be generated. This process is called data augmentation.

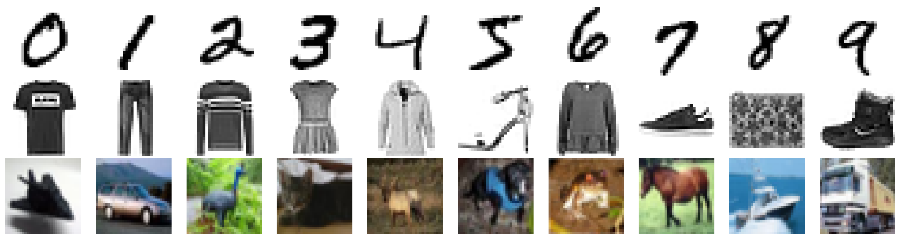

Figure 5.1: One sample of each class in the tested datasets. Each dataset contains ten classes. The first row shows the classes of the MNIST, the second the classes of the fashion-MNIST and the last row the classes of the CIFAR-10 dataset.

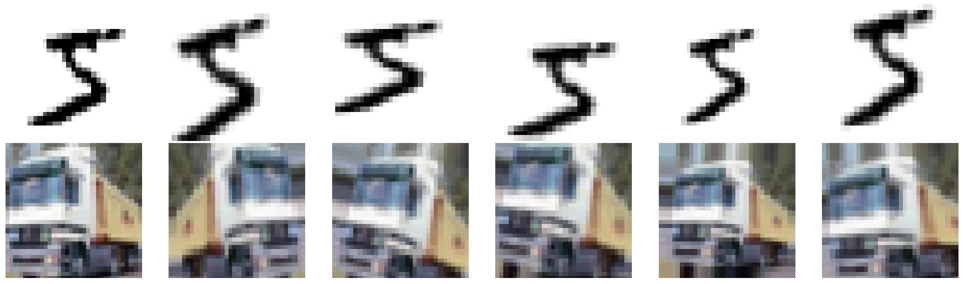

Figure 5.2: **Data augmentation examples** The first image of each row, shows the original, while the others are all generated through rotations, scalings and shifts of the original image. In the case of the truck image also horizontal flips were applied.

The goal of the data augmentation is to change existing data points without changing their classification. Common methods for the augmentation include scalings, rotations, shifts and other linear transformations. These transformations should be applied carefully, otherwise, the generated data points may be mislabeled. To set the correct parameters of the transformation domain knowledge plays an important part. For instance, an image of the digit eight can be horizontally flipped without changing the correct labeling. Flipping the digit five, on the other hand, creates a wrong labeled grid point.

Some valid transformations of an image from the MNIST and CIFAR-10 dataset are shown in Figure 5.2.

5.2 MNIST Handwritten Digit Dataset

The MNIST dataset consists of 60,000 grayscaled images of handwritten digits and additionally 10,000 validation samples. The classification goal is to decide which digit is shown in the image. Each image has a resolution of 28×28. The data hierarchy generated through the proposed general approach consists of six layers with the resolutions 28×28, 14×14, 7×7, 4×4, 2×2 and 1×1.

5.2.1 Paradox Behavior and the Dominance Effect

On the original dataset, without any data augmentation, I observed a, on first sight, paradox behavior: Sparse grids with lower levels have significantly better accuracy. This behavior can be seen in Table 5.1. The difference in the case without data hierarchy is more than 10%.

This behavior is probably caused by the dominance effect in the MNSIT dataset Waegemans described in [Wae17], as the confusion matrix shows that the sparse grid tends to label more samples to the dominant class if the level of the sparse grid is higher. The problem arises through a class that has, in general, a higher density over the complete input space. The SG++ software allows negative densities. Thus the dominance effect happens through local large negative densities for the dominant class, such that the positive density can be higher than the density of the other classes. A higher resolution of the sparse grid allows the density estimation to fluctuate stronger between negative and positive values. Therefore, in general, the more grid points the sparse grid consist of, the smaller the area with strong negative values can become so that the class can dominate all others in almost all areas. If a lower number of grid points is used, the areas with negative values can not switch back to a positive value, so that the dominant class has a lower density in more areas. If a refinement is applied on the sparse grid, the same effect is observable. After applying enough refinements all samples are getting labeled as the dominant class.

Since Waegemans traces the dominance effect of the classes back to their standard derivation [Wae17], I applied the data augmentation on the MNIST dataset. The data augmentation should increase the derivation and thus reduce the dominance effect. But instead, the results were worse. Even without applying any refinements, almost all samples were labeled to be samples from the dominant class.

Therefore, I changed the strategy. Instead of increasing the variance of the classes, I decreased it by calculating the average image of each class and using them as the training set. Thus the training set consists only of one sample per class. Surprisingly the results were actually nearly 3% better when taken over all classes. This may be caused by the reduction of data points as the density estimation will have stronger peaks at the local position of the one sample. As the samples were averages, these spikes were laying in good positions and reduced the dominance effect as they were more unlikely to overlap. Though, this did not solve the problems completely.

Therefore, the results and insights gained through the MNIST dataset have to be used carefully. For instance, the AHP-stencil turns out to have lower accuracy than the NHP-stencil. This may be caused by a lower accuracy of the AHP-stencil in general or it can also be caused by the paradox behavior as the AHP-stencil has more grid points.

5.2.2 Interpretation for the Data Hierarchy and Hierarchical Parent

Despite this paradox behavior, some interesting things can still be concluded from the results of Table 5.1.

First, the NHP-stencil reaches almost the identical accuracy than the DN-stencil without a data hierarchy and also decreases the number of grid points a little bit. As

stencil	level	grid points	train (%)	test (%)
Without data hierarchy				
regular sparse grid	2	1569	83.0193	82.398
DN	3	10753	74.7662	73.7812
With data hierarchy				
regular sparse grid	2	2101	83.3876	82.5113
DN	3	14253	74.9292	73.7812
NHP	3	10701	74.4616	73.7528
Cube-2 on {4×4, 2×2, 1×1} and NHP	3	11333	74.4262	73.441
DN and NHP	3	18653	73.1581	72.534
AHP	3	26353	62.6169	62.9252

Table 5.1: Classification results on the MNIST dataset using the classes 2, 5 and 7 on the resolution 28×28. Note that the accuracy of the sparse grids of level two is higher than their corresponding grids on level three. If no layers are specified, the stencil has been applied to all available layers.

the paradox effect can be ignored for results with a similar number of grid points and accuracy, it can be concluded that the NHP-stencil indeed does mimic the DN-stencil. Thus the NHP-stencil can be used to replace the DN-stencil and adding a data hierarchy without increasing the number of grid points, while still preserving a similar accuracy.

Secondly, adding the data hierarchy without exploiting it through applying a different stencil does not necessarily improve the accuracy. This can be seen by the identical accuracy of the DN-stencil of level three with and without a data hierarchy. Using a data hierarchy in this case only increased the number of grid points and thus the computing time.

5.3 Fashion-MNIST

Similar to the MNIST dataset is the fashion-MNIST dataset [XRV17]. It is meant as a drop-in replacement for the MNIST dataset. It has the identical structure and number of samples as the MNIST dataset but does not consist of images of digits. Instead, it consists of grayscaled pictures partitioned by ten different classes of fashion types. The classes are t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag and ankle boot.

This dataset was tested starting with a resolution of size 7×7. The used data

stencil	level	grid points	train (%)	test (%)
Without data hierarchy				
regular sparse grid	2	99	41.20	41.63
DN	3	631	38.03	37.88
With data hierarchy				
AHP	3	1417	43.04	42.78
DN and AHP	3	1865	40.56	40.47
regulr spare grid	2	141	40.10	40.27
NHP	4	3381	40.20	39.52
DN and NHP	4	5621	39.49	38.87
DN	3	869	38.31	38.09
DN	4	3221	38.40	37.49
DN and NHP	3	1349	36.99	36.62
NHP	3	901	36.48	35.97
DN on $\{4 \times 4, 2 \times 2, 1 \times 1\}$ and NHP	3	1013	36.32	35.84

Table 5.2: Classification results on the fashion-MNIST dataset using all ten classes on the resolution 7×7.

hierarchy added the coarsened layers of sizes 4×4, 2×2 and 1×1 to the data points.

As the results in Table 5.2 show, the fashion-MNIST dataset seems also to be having the paradox behavior. Although, the effect seems to be not as strong as on the MNIST dataset, as the difference between a level two and level three sparse grid without the data hierarchy is only around 5%. The improvement of the paradox behavior could be caused through the lower resolution of the images, as the dominance effect is highly depending on the standard derivation of the pixels [Wae17] and as the difference between the images becomes less on lower resolutions the effect is getting weaker.

Comparing the accuracy with and without data hierarchy, it can be concluded again that the data hierarchy by itself does not improve the accuracy of the classification. There is only a slight improvement of nearly 0.2%.

On this dataset, the NHP-stencil does not work as good as on the MNIST. There is a drop of around 2% compared to the DN-stencil without data hierarchy. On the other hand, the AHP-stencil does have very good accuracy, improving the best result without data hierarchy nearly 1% and if compared to the same level nearly 5%.

stencil	level	grid points	train (%)	test (%)
Without data hierarchy				
Cube-3	3	8941	31.75	31.86
Cube-2	3	4493	31.03	31.18
DN	3	3009	30.55	30.88
regular sparse grid	2	385	25.31	25.81
With data hierarchy				
DN and NHP	3	4899	30.80	31.17
ColDN and ColNHP	3	5239	30.42	30.66
Cube-2 starting from 4×4 and DN	3	4235	30.02	30.17
ColDN	3	4231	29.97	30.15
DN	3	3891	29.94	30.11
Cube-2 starting from 4×4 and NHP	3	5115	28.72	29.09
AHP	3	4267	28.36	29.07
ColNHP	3	3559	27.59	28.43
NHP	3	2539	27.35	28.13
regular sparse grid	2	511	25.28	25.48

Table 5.3: Classification results on the CIFAR-10 dataset using all ten classes on the resolution 8×8.

5.4 CIFAR-10

The CIFAR-10 dataset [KNH] consists of 50,000 training and 10,000 test images. The following labels should be assigned to the images: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

For the following test results, the original images have been coarsened to a resolution of 8×8. The images are colored with three color channels. The data hierarchy, generated by the proposed general approach, adds the layers with a resolution of 4×4, 2×2 and 1×1 to the data points.

5.4.1 Comparison between Different Stencils

The results generated on the dataset with different stencils are shown in Table 5.3.

The CIFAR-10 dataset does not seem to suffer from the paradox behavior and a look on its confusion matrix shows no sign of a dominant class. Therefore, as expected, the sparse grid of level two has a lower accuracy as the sparse grid of level tree. The increment of the level, in fact, improves the accuracy by 5%.

Simply adding the data hierarchy does, once again, not change the accuracy significantly. The decrement of the accuracy with data hierarchy using the same stencil as without is less than 0.8%.

Replacing the DN-stencil without a data hierarchy with a NHP-stencil reduces the accuracy by nearly 3%. Also, the AHP-stencil does not perform as well as on the fashion-MNIST dataset. These decreases of accuracy by the hierarchical parent stencils may be caused by the resolution. The resolution is of a power of two. Through this resolution, the data hierarchy can be divided evenly by two. Thus, each child has exactly one parent. The problem in such a case is that only a part of the neighbors of the child pixel is contained in the parent. Thus, the basic idea to replace the DN-stencil's interaction terms with an implicit interaction term to the hierarchical parent is not fulfilled and therefore the accuracy of the hierarchical parent stencils is not similar to the one of the DN-stencil.

Also interesting is that using the color versions of the stencils does not seem to affect the accuracy significantly. This may be caused by the low level, as the color interactions have interaction terms up to a size of three. Thus, only with a sparse grid higher than level four, the color versions of the stencils reach their full potential. On a sparse grid of level three and lower tough using them seems to only increase the number of grid points unnecessarily.

5.4.2 Different Layers

As the CIFAR-10 dataset does not have the paradox behavior, tests were made to inspect, how the used layers in the data hierarchy are contributing to the accuracy. Each test applied the DN-stencil and the NHP-stencil on the different layers. The results are presented in Table 5.4. Keep in mind that the NHP-stencil uses the next available layer even though it is not applied on it. Thus if it is applied on the layer 8×8, the layer 4×4 has been implicitly included.

As one can see, applying the stencils on the lowest layer results with the best accuracy. If the higher layers are included, the accuracy is reduced a little bit. This effect is probably caused by the coarsening algorithm used. As one can see in Figure 3.1 the coarsening reduces the coloring of the image until, in most cases at least, the image is only grayish. Thus, almost all samples are intersecting in this dimension. This increases the difficulty for the density estimation. This is also, probably why applying the data hierarchy with the same stencils as without it, may decrease the accuracy, as it can be seen especially for the sparse grids of level two in Table 5.3.

DN and NHP on	grid points	train (%)	test (%)
$\{8 \times 8\}$	3777	30.80	**31.17**
$\{4 \times 4\}$	903	28.00	28.13
$\{2 \times 2\}$	207	23.10	23.56
$\{1 \times 1\}$	33	20.37	20.99
$\{8 \times 8; 4 \times 4\}$	4673	30.65	31.06
$\{8 \times 8; 2 \times 2\}$	3977	30.44	30.72
$\{8 \times 8; 1 \times 1\}$	3803	30.78	30.97
$\{4 \times 4; 2 \times 2\}$	1103	27.38	27.46
$\{4 \times 4; 1 \times 1\}$	929	27.76	27.93
$\{2 \times 2; 1 \times 1\}$	233	22.97	23.32
$\{8 \times 8; 4 \times 4; 2 \times 2\}$	4873	30.44	30.84
$\{8 \times 8; 4 \times 4; 1 \times 1\}$	4699	30.61	31.03
$\{8 \times 8; 2 \times 2; 1 \times 1\}$	4003	30.42	30.59
$\{4 \times 4; 2 \times 2; 1 \times 1\}$	1129	27.15	27.37
$\{8 \times 8; 4 \times 4; 2 \times 2; 1 \times 1\}$	4899	30.37	30.75

Table 5.4: Classification results on the CIFAR-10 dataset in correlation with different layers. With the NHP-stencil being applied on a certain layer, it is meant that interaction terms between the pixels in the specific layers and the next hierarchical parent are added. Thus, if the stencil is applied on the layer with resolution $\{8 \times 8\}$, the pixels from the layer with resolution $\{4 \times 4\}$ are included as well.

6 Conclusion

The data hierarchy allows the use of complex stencils on lower resolutions while applying simple stencils on the original data and thus solves the tradeoff that must be made between a higher or lower resolution. Since better stencils can be applied, this allows improving the image classification accuracy. Although, simply using the same stencils as in the case without the data hierarchy does not seem to increase the accuracy necessarily. In such cases, some layers of the data hierarchy should be left out, as the layers with lower resolutions tend to get the same values through the coarsening technique and thus may decrease the accuracy. The data hierarchy also enables us to use new stencil types, for instance, the HP-stencils.

The NHP-stencil seems to mimic the accuracy of the DN-stencil quite well and reduces the number of grid points. It tends to work better between layers that are not dividable by their resolution. Therefore, in such cases, the generation of the data hierarchy should be modified a little bit to always end with undividable resolutions.

As the changes of accuracy in the classification results range from -2% to 5%, using a data hierarchy may be useful in some cases, but only if it is good optimized for the specific problem. Further investigation is needed, to exploit a data hierarchy reliably.

7 Future Research

Further research could develop a wider variant of stencils that work vertically on the data hierarchy. As already mentioned in the classification results the NHP-stencil seems to have a decreased accuracy when the resolution is of a power of two. This is probably because the parent pixel does not include all neighbors of the current pixel. Instead, it only contains a corner of the neighbors, since four pixels without any overlappings are contained in the parent, if the data hierarchy is generated through the proposed general method. An on the data hierarchy vertical working stencil, that works similar to the hierarchical-parents stencils, but includes interactions to the neighbors of the hierarchical children as well may increase the accuracy.

Another interesting stencil would be one that includes all hierarchical parents into a single interaction term. This will increase the number of grid points but it would be interesting to see, how the accuracy could be improved. Intuitively this approach compares the different sizes of the area of a pixel. It could better detect small details, for instance, if a green pixel is in a white area, that itself is in a red area and so on.

Additionally, it could be interesting to investigate how the accuracy is affected by a refinement on a sparse grid with and without data hierarchy. Maybe through refinement steps the sparse grid can make better use of the additional data dimensions.

Last but not least, it is important to further investigate the paradox behavior on the MNIST and fashion-MNIST dataset. As the MNIST dataset is one of the most famous datasets for image classification, it is important to understand why the sparse grid density estimation or the SG++ library is failing on it.

8 Appendix

8.1 Confusion Matrices

	2	5	7
2	791	19	371
5	3	524	531
7	1	0	1288

Table 8.1: Confusion matrix for a MNIST test on a resolution 28 × 28.

	0	1	2	3	4	5	6	7	8	9
0	154	829	5	94	23	2	2	13	0	0
1	0	1188	3	8	18	2	0	1	0	0
2	6	258	133	33	709	33	2	24	2	1
3	2	1150	0	49	10	0	0	1	0	0
4	0	312	12	80	758	9	0	10	0	0
5	0	9	0	0	0	355	0	818	0	22
6	45	530	43	101	454	34	3	30	3	1
7	0	0	0	0	0	3	0	1180	0	9
8	5	111	11	15	35	60	0	887	34	37
9	0	13	0	0	0	38	0	461	0	717
percentage (%)	1.77	36.67	1.73	3.17	16.73	4.47	0.06	28.54	0.33	6.56

Table 8.2: Confusion matrix for a fashion-MNIST test on a resolution 7 × 7.

8.2 Miscellaneous

```
{
    "dataSource": {
```

	0	1	2	3	4	5	6	7	8	9
0	386	55	6	16	39	7	70	18	274	134
1	44	196	3	20	54	6	227	22	88	314
2	168	39	38	31	216	5	331	59	63	82
3	89	33	13	77	111	69	380	54	40	150
4	63	13	13	26	340	14	371	45	38	76
5	62	24	9	69	144	138	293	62	54	82
6	44	22	15	26	153	22	606	46	15	81
7	53	20	7	42	183	25	240	170	40	221
8	98	78	2	21	38	21	63	11	478	215
9	44	58	3	3	34	7	120	22	108	582
percentage (%)	10.51	5.38	1.09	3.31	13.12	3.14	27.01	5.09	11.98	19,37

Table 8.3: Confusion matrix for a CIFAR-10 test on a resolution 8×8.

```
        "filePath": "cifar10.arff",
        "hasTargets": true
    },
    "fitter": {
        "gridConfig": {
            "gridType": "modlinear",
            "level": 3
        },
        "geometryConfig": {
            "dim": [
                [32,32,3],
                [16,16,3],
                [8,8,3],
                [4,4,3],
                [2,2,3],
                [1,1,3]
            ],
            "stencils": [
                {
                    "stencil": "DirectNeighbour",
                    "applyOnLayers": [0]
                },
                {
```

```
                "stencil": "NextHierarchicalParent",
                "applyOnLayers": [0]
            }
        ]
    },
    "regularizationConfig": {
        "regularizationType": "identity",
        "lambda": 0.01
    },
    "densityEstimationConfig": {
        "densityEstimationType": "decomposition",
        "matrixDecompositionType": "chol"
    }
  }
}
```

Listing 8.1: Example SG++ Pipeline Config

Bibliography

[BG04] H.-J. Bungartz and M. Griebel. "Sparse grids." In: *Acta Numerica* 13 (2004), pp. 147–269. ISSN: 0962-4929. DOI: `10.1017/S0962492904000182`.

[Bra00] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[Cho+15] F. Chollet et al. *Keras*. `https://keras.io`. 2015.

[KNH] A. Krizhevsky, V. Nair, and G. Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)." In: ().

[Kre16] L. Krenz. "Integration of Prior Knowledge for Regression and Classification with Sparse Grids." Bachelor's thesis. Institut für Informatik, Technische Universität München, Aug. 2016.

[Mar18] B. Marr. *How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read*. 2018.

[Peh13] Peherstorfer, Benjamin. "Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques." In: (2013).

[Pfl10] D. Pflüger. *Spatially adaptive sparse grids for high-dimensional problems: Zugl.: München, Techn. Univ., Diss., 2010*. 1. Aufl. München: Verl. Dr. Hut, 2010. ISBN: 9783868535556.

[PPB10] D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. "Spatially adaptive sparse grids for high-dimensional data-driven problems." In: *Journal of Complexity* 26.5 (Oct. 2010). published online April 2010, pp. 508–522. ISSN: 0885-064X.

[vCV11] S. van der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy Array: A Structure for Efficient Numerical Computation." In: *Computing in Science Engineering* 13.2 (Mar. 2011), pp. 22–30. ISSN: 1521-9615. DOI: `10.1109/MCSE.2011.37`.

[Wae17] Waegemans, Tim. "Image Classification with Geometrically Aware Sparse Grids." In: (2017).

[XRV17] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: `cs.LG/1708.07747 [cs.LG]`.