Awarding institution : TU Delft
Supervisor : Dr. David M. J. Tax
Research group: Pattern Recognition & Bioinformatics
Thesis Committee : David M. J. Tax, Prof. Marcel J. T. Reinders, Dr. Christoph Lofi

MSc thesis in Computer Science

Slicing methods for deep learning on volumetric data

Ynze ter Horst

May 2023

Abstract

Computer vision applications for two-dimensional images and video are widespread nowadays, but three-dimensional imagery has been a less common modality in the field of deep learning. Volumetric data in relation to planar data introduces a third spatial dimension, increasing both the information entropy and dimensionality of datapoints. Conventional models trained on such data are required to address the additional input dimensionality with a larger parameter space. This makes both evaluating and training them more computationally complex, especially in terms of spatial complexity. Slicing architectures try to tackle these issues by considering volumes to be a collection of slices stacked across one of three slicing dimensions. They reduce the slices in isolation and combine them in a compact combination stage. This approach allows for advantages like parallelization, generalization and computational reduction. The slicing strategy assumes however, that the slices are identically and independently distributed. The extent to which this assumption is valid, is dependent on context and slicing dimension. This thesis introduces a range of deep learning architectures that use slicing and conventional approaches, to uncover the relation between them in the context of artificial and medical volumetric data. It will investigate the utility of slicing methods as an addition to autoencoder models and illustrate the importance of the slicing dimension.

1. Introduction

The field of computer vision has originally dealt with two-dimensional images. The first groundbreaking papers that demonstrated the utility of deep learning networks, like ImageNet [2] and AlexNet [5], used two-dimensional convolution to classify three-channel images. Some domains however, are concerned with volumetric data. The medical domain for instance, often captures data with a single channel representing tissue densities measured using resonance (in the case of ultrasound) or absorption (in the case of computed tomography and magnetic resonance imaging). These volumes are laid out in grids with three dimensions according to some resolution, each corresponding to one dimension in space.

An advantage of such an additional dimension is the information it gives about depth. Standard images are highly dependent on the real-world perspective of the camera plane on which the scene is projected. This makes them subject to ambiguities like occlusion, background clutter and scale. With three-dimensional data, voxels (volumetric pixels) can be scaled to a constant volume, allowing for a more robust representation of real-world space.

Secondly, it simply contains more information. Correlations are usually large across any of the spatial dimensions, including the third 'depth' dimension. Even though this will make for low information density given the dimensionality of the volumes, a single observation will have higher entropy than a single two-dimensional image capture of the same class.

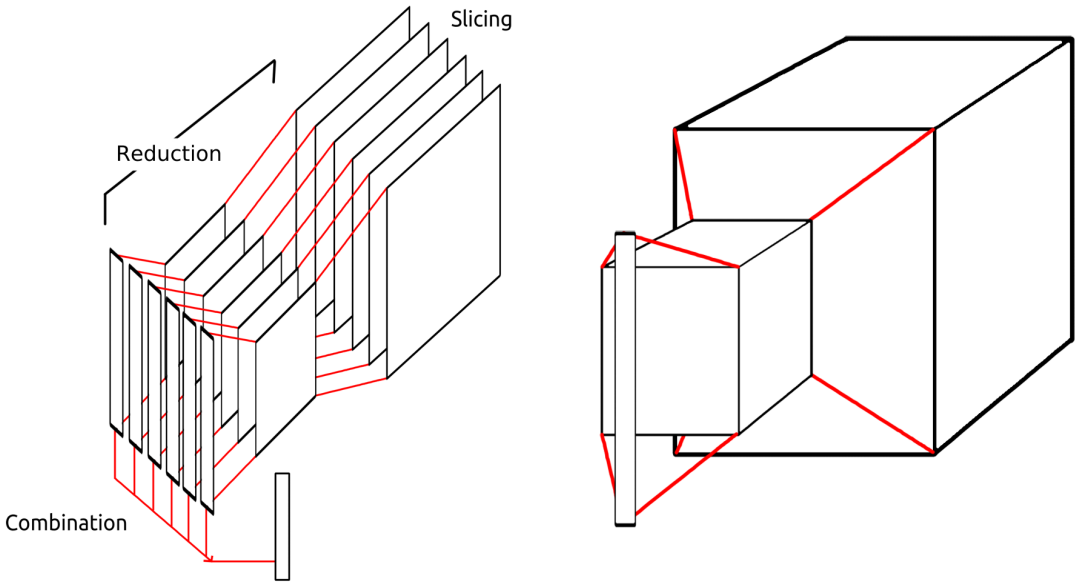
Unfortunately, three-dimensional data comes with issues as well. Most importantly, the amount of input features grows significantly. If we assume a square image of input dimensionality $resolution^2 = n$, a corresponding cubic volume will grow by a factor in the order of $\sqrt{n}$ (since $n * \sqrt{n} = resolution^3$) with respect to the two-dimensional imagery. This is an issue as most operations in deep learning applications, like convolution and data loading, have O(n) space and time complexity. Meaning required computational resources will scale linearly with the square root input growth.

This only considers the deep learning model as a function itself, not the learning of the function. The third dimension radically enlarges the gradient tree traversed during back-propagation. The combination of larger input and parameter spaces makes the differentiation process much more computationally complex.

Large parameter spaces additionally are prone to overfitting and make models reliant on other regularization methods for generalization.

1.1. Slicing

This is where slicing methods may provide some solutions. The idea behind them is the separation of one of the spatial dimensions into single slices, that are treated separately in the first stage of some deep learning process. The slices are individually reduced to a lower dimensional representation using a learnable function that is independent of the position in the slicing dimension, and combined in a later stage across the slicing dimension. Figure 1.1 shows a schematic comparison between conventional methods and slicing methods. We will sometimes refer to the slicing approach as the two-dimensional approach, as opposed to the



(a) Schematic of the slicing approach.

(b) Schematic of the three-dimensional convolutional approach.

Figure 1.1.: Visualization of compressing volumes to low dimensional vectors, using slicing or three-dimensional convolution.

three-dimensional approach.

The primary benefit that comes with this approach is concerned with memory. Breaking up the dependencies in one of the three spatial dimensions allows both the forwarding of input and the backpropagating of output through the neural network to consider slices in isolation. Only the second stage of combining the representations - which is in the same order of computation as reducing a single slice - requires all intermediate results. In the context of processing large volumes with limited memory resources, like in embedded systems, slicing may be crucial for feasibility.

This notion of isolated slices also gives the network a parallelization property, allowing it to be trained and evaluated in arbitrary order of slice before the combination step. If one would like to produce dedicated hardware that could for instance produce diagnostics immediately after capturing medical images in scanning machines, this property can be exploited to optimize parallelizable operations. Alcaín et al. [1] discuss performance and power efficiency at the expense of unitary cost of such dedicated chips in medical imaging machines. Appendix B.2 gives a more detailed description of parallelizability of backpropagation.

Some volumes, like CT scans in numerous cases, specifically have one dimension with variable resolution. In the three-dimensional setting one would need to resort to pooling operations to account for this variability, which eliminate a large part of the spatial information. In the two-dimensional setting however, a slice combination method can be used that does not depend on a predetermined amount of slices. A temporal model like an RNN could for instance be employed, that interprets the last spatial dimension as a temporal one.

A final advantage of slicing is the generalization encoded in the neural structure. By considering a third dimension to be able to be encoded without considering the spatial information

across that dimension, the network is forced to learn the dependencies in the dimension only later in a compact combination stage. As a result the network becomes less complex in terms of its parameter space and target function.

A trade-off to consider though, is that slicing considers 3D volumes to be collections of 2D images stacked in one of the three spatial dimensions, and drawn from the same distribution. This to some extent violates the i.i.d. (independently and identically distributed) assumption. Slices will not be entirely independent as there will be strong correlations in the third spatial dimension. Additionally, the underlying distribution we draw from is partly a function of the position in the slicing dimension.

In other words, there is information contained in the last spatial dimension that is lost during slicing. The question is to what extent the loss of this information compromises the ability of a model to learn the topological combination of the lower dimensional representations in a later stage.

This thesis will investigate how the performance of two-dimensional slicing methods compare to three-dimensional convolutional methods in the classification of a target corresponding to volumetric data. We will present a number of architectures, for both strategies and two input space sizes corresponding to different volumetric contexts. They will be trained and evaluated on an artificial dataset containing geometric objects and a dataset containing a limited number of ultrasound scans of uteri in the first trimester of pregnancy. Additionally, we will introduce a 'slicing autoencoder' design that combines the slicing principle with self-supervision. An extensive discussion on the impact and choice for the slicing dimension is included as well.

1.2. Related work

In the last decade deep learning approaches in the volumetric domain have made their appearance, both in the slicing and three-dimensional setting. Some of the literature concerning these architectures present promising results.

In [8], a placenta segmentation network for three-dimensional ultrasound scans is introduced. The network, DeepMedic, is trained on only 300 ultrasound scans, and achieves a good segmentation performance relative to manual ones. However, the model does suffer from rotational and positional variance of the implantation site of the placenta in the uterus. A paper by Lange et al. [6], uses an autoencoder architecture to recontruct individual slices of MRI scans of the human liver. One of the findings of the paper is that spatial information has to be retained in the latent space to allow for reconstruction of the images. Some papers admit that three-dimensional image data is often too high dimensional to consider as a whole, especially when only a small number of volumes are available. They propose a slicing approach, which combine encodings using gaussian methods [20] or convolution [15]. Yamaguchi et al. [23] think crucial spatial information in the third dimension is lost in that slicing process, so a three-dimensional convolutional autoencoder is consulted instead in the classification of schizophrenia. Martinez et al. also use a simple three-dimensional autoencoding architecture in combination with a SVM to classify Alzheimers disease using MRI images of the brain [9]. Oh et al. [12] take a similar approach, but with a pretrained convolutional autoencoder.

Other literature takes a more sceptical stance towards deep learning for three-dimensional scans.

Shen et al. [16] challenge the interpretability and flexibility of deep learning models in the medical domain. Different modalities require different domain specific architectures and will

always be dependent on large datasets. Diniz et al. [3] target ultrasound scans specifically. One statement, "Large, public and appropriately quality-controlled ultrasound datasets are needed to compare different deep learning methods and achieve robust performance in real-world scenarios.", captures a number of issues with three-dimensional ultrasound data. It also mentions a lack of quality in the data and advises to test new deep learning methods on homogeneous medical imaging methods like MRI or CT scans before experimenting with ultrasound data. Locatello et al. [7], conclude that disentangled representation learning is fundamentally impossible without inductive biases in the learning approach and dataset. To guarantee validity of results, experiments should always be conducted in a reproducible setup and on datasets of varying degrees of difficulty.

Some of these setups take the additional complexity of the three-dimensional convolution for what it is and train on entire volumes nonetheless. These methods are often concerned with low resolutions, making them feasible for training on a conventional small-scale deep learning setup. Unfortunately, real-world volumetric domains are rarely concerned with resolutions as low as these and do not allow for acceptable training batch sizes.

2. Methods

To evaluate the relevance of slicing methods, we have constructed a number of experimental setups. Each of these is concerned with a deep neural network aimed at a specific input space that does or does not use a slicing strategy. In this section we will go over the details of these experimental setups and the choices for their corresponding architectures.

All experiments were done on a HP ZBook G4 with an Octacore Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz processor and NVIDIA Corporation GM107GLM [Quadro M1200 Mobile] graphics card.

2.1. Architectures

In terms of architecture, six different setups are considered. Two of these use a slicing strategy in either a small or large input space, two use a three-dimensional convolutional approach in the same large and small input spaces. The last two are autoencoding architectures combined with slicing or 3D convolution, that operate in the large input space as well. Table 1 summarizes the key differences.

We will first go over the architectures for small input dimensionality extensively, to illustrate the differences between slicing and 3D approaches. Figures A.3 and A.4 show the layering of these architectures.

Batches are first normalized as an additional preprocessing method to tackle some of the input variance. The compression stages are identical in the kernel and channel sizes between slicing and 3D convolutional approaches in the same context, but operate either in 2D or 3D space depending on their strategy. Consequently, the 3D version requires additional parameters for the third kernel dimension. This compression stage consists of five convolutional blocks, comprised of a ReLU non-linearity and maxpool operator in addition to the convolution itself. A second batch normalization layer is employed at this time to counter vanishing output variance. In the slicing approaches slices are now combined using a convolutional kernel that interprets the concatenated slice representations as n channels to be mapped to a single channel, where n is the resolution of the slicing dimension. This function can be thought of as a linear combination of the compressed representations across the last spatial dimension.

Inputs in both 3D and 2D settings have now been reduced to vectors with the same dimensionality, so we can use identical subnetworks to classify geometric shapes or regress gestational age in some downstream process. In the former we target a four-dimensional class probability vector and in the latter a single scalar. This is done with three linear layers and non-linearities, to allow the network to propagate spatial information. For the classification network a final softmax layer is used to normalize the probability vector.

The architectures aimed at the larger input space simply incorporate a number of extra convolutional and pooling operators in addition to the previous architecture to compress the higher dimensionality, so they will not be described in detail in this section. Their schematic visualization can be found in figures A.5 and A.6.

Architecture Comparison						
	A.3	A.4	A.5	A.6	A.7	A.8
Data	Geometry	Geometry	Ultrasound	Ultrasound	Ultrasound	Ultrasound
Input size	32^3	32^3	256^3	256^3	256^3	256^3
Output size	4	4	1	1	32 (latent)	32 (latent)
Problem	classification	classification	regression	regression	autoencoding	autoencoding
Approach	3D convolve	Slicing	3D convolve	Slicing	3D AE	Slicing AE
Parameters	9,514	4,439	21,927	9,140	65,029	12,854

Table 1.: Comparison between architectures.

The autoencoding architectures are a bit more involved. A definition of autoencoding can be found in appendix B.1, and the corresponding architectures in figures A.7 and A.8. Both architectures reduce input to a 32-dimensional latent vector before decoding to the initial input dimensionality.

In the slicing variation, the input space is once again encoded slice-wise and combined into a latent vector at a later stage. In the decoding stage, an additional kernel is used to learn the inverse mapping of the latent vector to a sequence of low dimensional feature vectors for each position in the slicing dimension. The vectors at these positions are then decoded in isolation to the complete input space and compared with initial input.

Both encoder and decoder subnetworks have channel-wise fully connected layers. These can be thought of as intra-channel fully connected layers. No information is propagated across channels, but all spatial information inside of a single channel is combined. This greatly reduces complexity while only compromising slightly on expressibility. Assuming identical feature map sizes for input and output of the c channels, taking a channel-wise approach always reduces the amount of parameters by a factor c. A paper by Pathak et al. [13] introduces these layers in autoencoders that encode an omitted area of a picture using its context.

2.2. Regression and classification

Both datasets introduced in section 3 were split up in training and test sets. Networks are re-evaluated on the independent test set between training iterations, to produce a test curve throughout the learning process. All experiments use an Adam [4] optimizer with decaying learning rate starting at 0.01. Batch sizes were 32 and 4, for small and large architectures respectively. These batches are sampled randomly, but always have uniform class priors in the case of the geometry dataset (class balancing). The mean squared error loss measures the performance of regression and classification. For the geometry classifiers early stopping is implemented below a MSE loss of 10^{-3}.

2.3. Slicing autoencoders

The slicing approach we have dsicussed can also be applied to autoencoders, by encoding and decoding individual slices using the same neural network, and combining them in the last spatial dimension only in the bottleneck layer. This leaves us with all the benefits we have talked about, and all the drawbacks as well. Hopefully however, the combination with an autoencoding architecture gives the model an additional regularization property that allows

it to better predict gestational age. The autoencoders use all the same hyperparameters introduced in the previous section, but MSE loss is calculated between reconstruction and input.

2.4. Slicing dimension

When a slicing method is applied, there is an additional choice to be made for an important hyperparameter. Which of the three dimensions will be the slicing dimension? Intuitively, we would like our slicing dimension to break up as little information as possible. Particularly, we expect the dimension along which correlations are largest to be the optimal slicing dimension. When slicing this dimension, the i.i.d. assumption will be most valid as the sampling distribution is most similar.

We will present a simple algorithm to find such a dimension. First take a dimension along which you would like to know the slicing impact. Stack all available volumes along this dimension. A dataset with m volumes of average resolution r in the slicing dimension will now give you $n * r$ slices that all have the same two spatial dimensions. Create a band of slices of size k, where k is the kernel size of the compared 3D convolutional method. Flatten the slices (pixel-maps of $n * n$) to vectors (of size n^2), and calculate correlations between the left-most and right-most slice in the band. Now iterate the band through the slicing dimension to get a collection of correlations. In turn, average these for an approximation of the correlation across the spatial dimension. Repeat this process for all three dimensions and compare the different impacts of slicing.

In this report we will additionally take an experimental approach to finding such a dimension. Both datasets and two transposed versions of each of them were created. The transposed versions project the second and third spatial dimension onto the slicing dimension. The slicing architectures were trained and evaluated on their respective datasets and transposed versions a number of times, to see if either dataset significantly benefitted from a specific choice for the slicing dimension.

To investigate if the slicing dimension has an effect on the type of errors models make, confusion matrices are built up in the geometry classification process for different slicing dimensions as well.

3. Data

To conduct comparative experiments we have considered two volumetric datasets in the evaluation of slicing methods. Their context may seem more different than related but they represent different challenges with respect to noise, dimensionality, variance, size and other properties in the same context of deep learning on volumetric data.

Dataset Comparison		
	Artifical Geometry	3D Ultrasound
Dimensions	32^3	256^3
Noise	noiseless	noisy
Size	2000 samples	412 samples
Heterogeneity	translation and rotation	translation and rotation
Target	shape	gestational age
Sparsity	sparse	very sparse
Context	artificial	medical observation
Scale	constant	normalized

Table 2.: Comparison between considered datasets in a number of relevant characteristics for the evaluation of slicing methods.

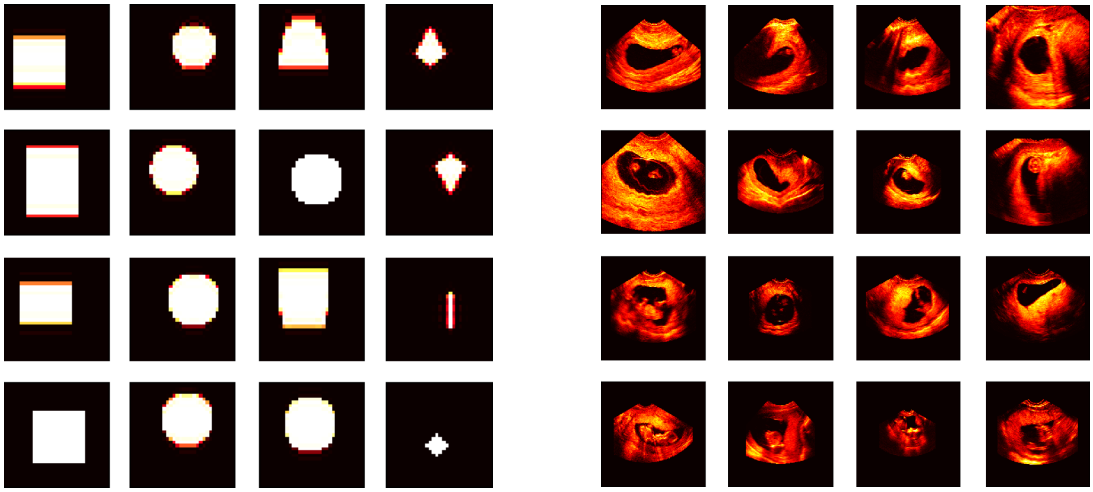
3.1. Artificial geometry

The artificial geometry dataset can be thought of as simple geometric shapes in a cubic volume with resolution 32^3. This dataset is used to test slicing in a controlled context, so we can create a setup that is a precursor to a model aimed at more troublesome datasets. Is it possible to create a reasonable shape classifier with the slicing and 3D architectures in a noiseless space with relatively small dimensionality?

A subvolume of 16^3 is filled with one of four objects; a sphere, a cube, a cylinder or a tetrahedron. The idea behind these choices for objects is the fact that without a combination of spatial information in all three dimensions, ambiguities between shapes start to arise. Specifically cylinder-sphere and cube-tetrahedron pairs will be hard to tell apart without the depth dimension.

To simulate input variance the subvolumes containing geometric shapes can occupy one of five positions in each of the three dimensions, leaving us with a total of 5^3 different translated versions of a single object. All four shapes are translated to all positions to preserve class balance.

Additionally, an object is rotated to one of four orientations. Specifically, $\{0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}\}$ is the set of equally probable rotations in the plane of what we will call the first two spatial dimensions. Consult figure 3.1a for a collection of examples.



(a) Examples of middle slices of artificial geometry. Cubes, spheres, cylinders and tetrahedrons; in order of column, with random translation and rotation.

(b) Examples of middle slices of ultrasound scans. Rows from top to bottom correspond to gestational ages 7, 9, 11 and 13 weeks respectively.

Figure 3.1.: Examples of data.

Leaving out a number of combinations of these rotations and translations and using them as test data would mean the set is comprised of linear combinations of training data. To make the set more challenging, all translation-rotation combination are preserved in the training set while the test set is filled with interpolations.

3.2. First trimester intrauterine ultrasound

We also consider a dataset with three-dimensional ultrasound scans of first trimester uteri. In comparison with the artificial geometry dataset, it is much more problematic for a number of reasons. It is high in its dimensionality with a resolution of 256 in each of the three spatial dimensions. There is a limited number of samples because of reliance on volunteers that are comfortable with the privacy sensitivity of the data. The data has high variance; rotational and translational heterogeneity. Additionally, classes are generally imbalanced and some of the targets we would be interested in predicting are even nearly nonexistant. Last but not least, there is inherent and substantial noise in the data due to the mode of capture being high frequency sound.

The scans we will use are what we call 'raw' scans. They are volumes with density values as retrieved directly from the ultrasound machine. These scans are sparse and a large part of their volume contains useless information. Their benefit is the absence of variance as a result of human bias and pre-processing methods, in contrast with heuristic tempering and human segmentations of the scans.

The collection of this dataset is part of the Rotterdam Predict program [14]. It is collected in a hospital-based cohort study by the Erasmus Medical Center in Rotterdam, the Netherlands. Intrauterine 3D ultrasound scans of embryos are taken at multiple stages throughout the first

trimester of pregnancy, using the Voluson E8 and E10 by GE healthcare. A large range of medical and behavioral markers are also available, some of which doctors would be interested in predicting or correcting for. In this study we will consider the gestational age in weeks, the amount of time since the conception of the embryo. It is hypothesized that a robust model for gestational age may be able to use outliers as a measure for growth issues during pregnancy and/or predict birth weight. Figure 3.1b contains a number of examples of middle slices of ultrasound scans.

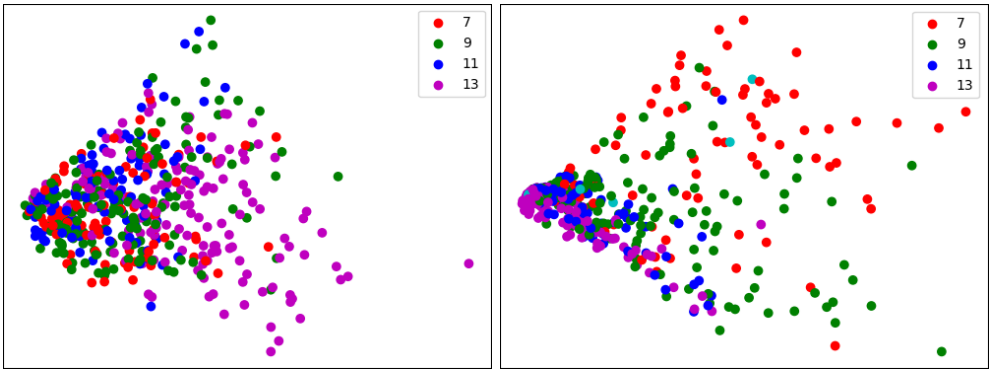
Preprocessing

Unlike the artificial dataset, the ultrasound dataset required a number of preprocessing steps before being eligible for training on targets.

First and foremost, the volume of voxels in scans are not standard, but vary per scan. Fortunately, the machine that captures the scans records the length of all dimensions in millimeters in the metadata. To be able to exploit the previously mentioned benefit of capturing scale using a third dimension in space, the voxels first needed to be projected to a space with standard dimensions. This was done using spline interpolation.

Figure 3.2 illustrates the effect of normalization. Normalized and non-normalized ultrasound scans are projected onto the first two principal components of the latent space produced by the autoencoder in figure A.7. The colors represent gestational age targets corresponding to the scans. It is clear that normalization significantly increases separation.

Since the non-normalized voxel sizes deviate in a large range, scaling can make any of the dimensions either too small or too large in resolution. The simple approach we took to alleviate this issue is the padding and slicing in all dimensions until the required resolution is attained. Slicing and padding maintains center, allowing our spatially variant architectures to learn global features.



(a) Before voxel normalization. (b) After voxel normalization.

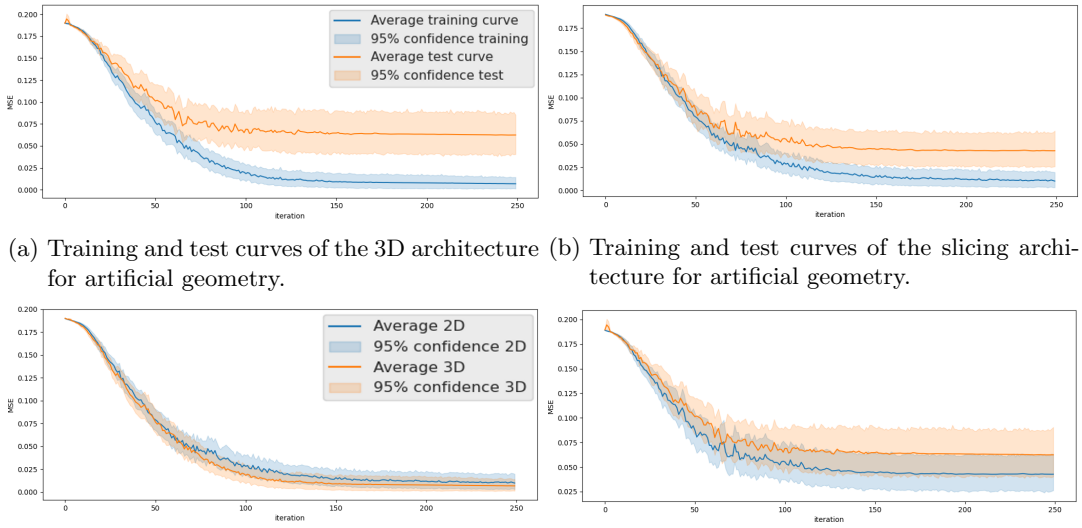
Figure 3.2.: Illustration of the effect of voxel normalization using the first two principal components of the latent space of the 3D autoencoder. Color-coded for gestational age targets in weeks.

4. Results

In this section we will present our findings in the experiments outlined in section 2. The results consist of average learning curves and confidence bands in a learning process using the presented architectures and corresponding datasets. Additionally, final average test performances are compared in a range of experiments surrounding noise, slicing dimension choice and autoencoding.

Because experiments are stochastic in batch sampling and network initialization, all statistics are aggregations of a number of runs. Means, standard deviations and number of reruns are mentioned with results.

4.1. Artificial geometry



(a) Training and test curves of the 3D architecture for artificial geometry.

(b) Training and test curves of the slicing architecture for artificial geometry.

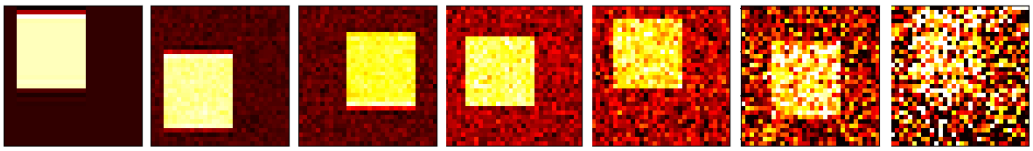
(c) Training curve comparison for slicing and 3D convolutional approach.

(d) Test curve comparison between slicing and 3D convolutional approach.

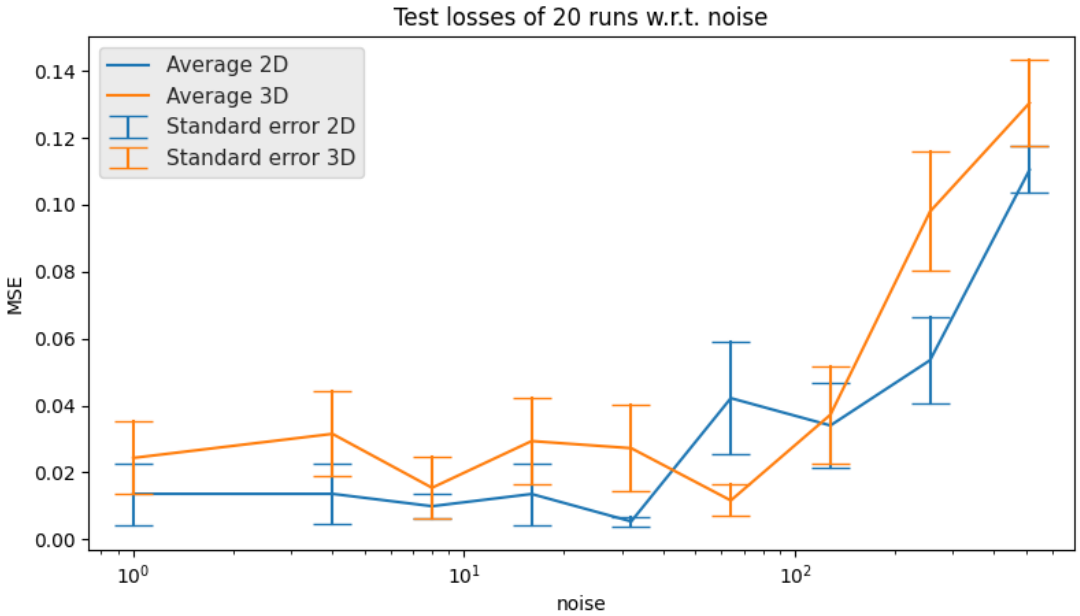
Figure 4.1.: Comparison between slicing and 3D convolutional approaches, training and test curves in the artificial geometry setting. Experiments were rerun 50 times.

The geometry dataset is concerned with classifying one of four artificial geometric objects, using architures A.3 (3D convolution) and A.4 (slicing). The results in figure 4.1 show the one-hot classification MSE mean and 95% confidence intervals of fifty reruns of the same experiment.

In figure 4.1a, training and test curves seem to be separated by a large margin. In figure 4.1b however, the confidence bands are closer together. In other words, performance on the



(a) Visualization of middle slices of random squares in increasingly noisy environments.



(b) Test score comparison between strategies in corresponding noisy environments. Noise is measured in standard deviation of the added gaussian.

Figure 4.2.: Test score comparison between strategies in noisy environments. Noise is measured in standard deviation of the added gaussian.

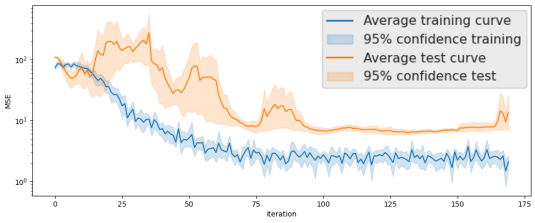
training set is too optimistic in the case of the 3D setting. It outperforms the slicing setting on the training set as can be seen in 4.1c, still slicing outperforms 3D convolution on the test set. Variances do not seem to be strategy-dependent.

The slicing setting allows less overfitting on the training set, this is to be expected because of smaller model complexity and spatial generalization. The 3D approach can exploit more information in the slicing dimension, giving it a better training score, but a lack of generalization. The training process is not significantly faster in the 3D setting but it converges to a better solution after the additional parameter complexity starts to be exploited.

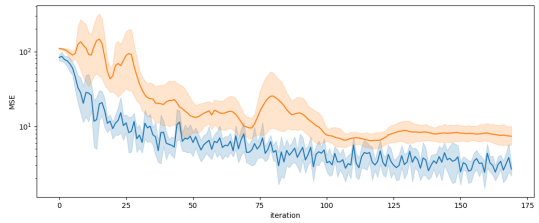
To see if added noise would change the relationship between test performances, an increasing amplitude of gaussian noise $G \sim N(\mu = 0, \sigma = noise)$ was added to the data. Figure 4.2 shows a comparison of test scores in increasingly noisy environments.

Even with twenty reruns, MSE deviations are large and standard error intervals overlap. The slicing strategy scores a lower mean almost consistently and variances do not seem to be significantly affected as noise starts to increase in the volume. Both strategies suffer from added noise, but only after σ becomes substantial.

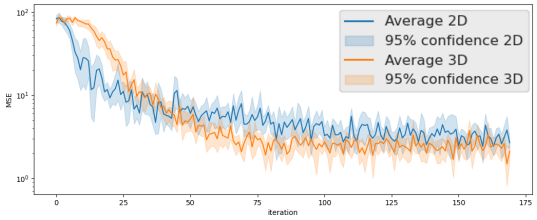
4.2. Ultrasound scans



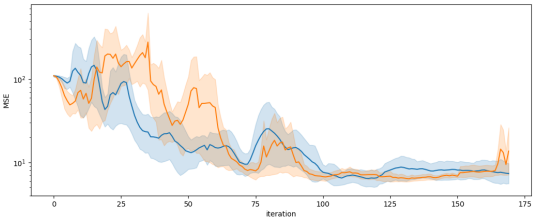
(a) Training and test curves of the 3D convolutional architecture for ultrasound pregnancy scans.



(b) Training and test curves of the slicing architecture for ultrasound pregnancy scans.



(c) Training curve comparison for slicing and 3D convolutional approach.



(d) Test curve comparison between slicing and 3D convolutional approach.

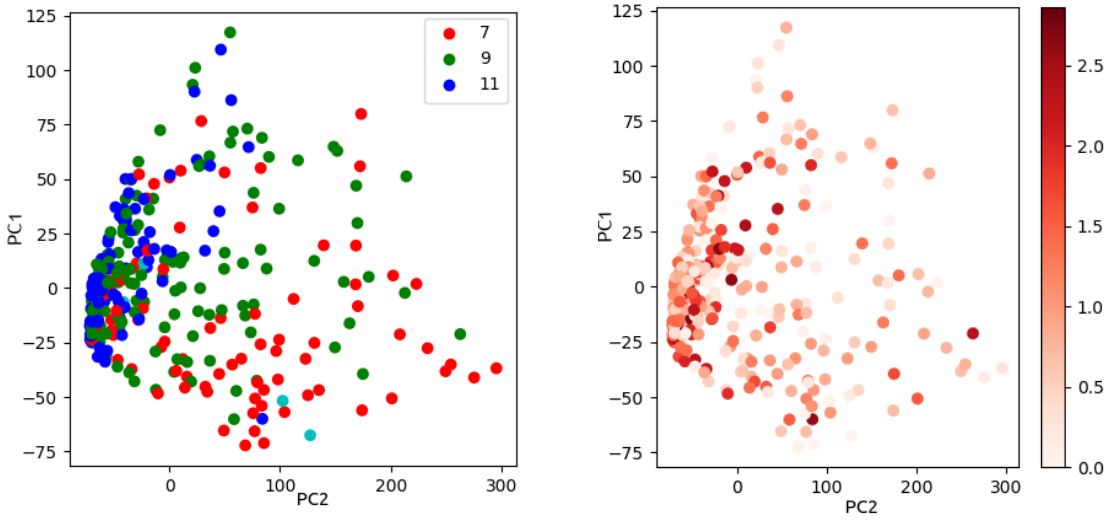
Figure 4.3.: Comparison between slicing and 3D convolutional approaches. Training and test curves in the ultrasound scan setting, log scaled. Experiments were rerun 10 times.

The architectures for the ultrasound data (A.5 and A.6) try to predict gestational age targets. Figure 4.3 shows mean and 95% confidence intervals of MSE regression loss on logarithmic scale.

In the context of ultrasound scans, overfitting is an issue for both approaches. This seems to be related to the small batch size feasibility relative to the large input space. Training of the more complex 3D architecture takes additional iterations, but test performances do not indicate radically different convergence properties. Solution qualities are similar, despite differences in parameter space sizes and inductive prior of the slicing method. Stochasticity has a larger effect on the 2D architecture, whereas the variance is lower throughout testing the 3D architecture. Arguably, slicing is more prone to stochasticity in the sampling of small batches because of non-correlations between samples across the slicing dimension. Test curves are smoother because of large test set size compared to batch sizes.

4.3. Slicing autoencoders

Both autoencoders were first trained on the same training set. Figure 4.4a visualizes the latent space produced by a trained autoencoder. After training, an analytical linear regression solution was fitted to the 32-dimensional latent space of both. Figure 4.4b visualizes regression errors in a random training iteration. This process was repeated 6 times. The conventional autoencoder achieved a final average test score mean of 1.0117, where the slicing autoencoder



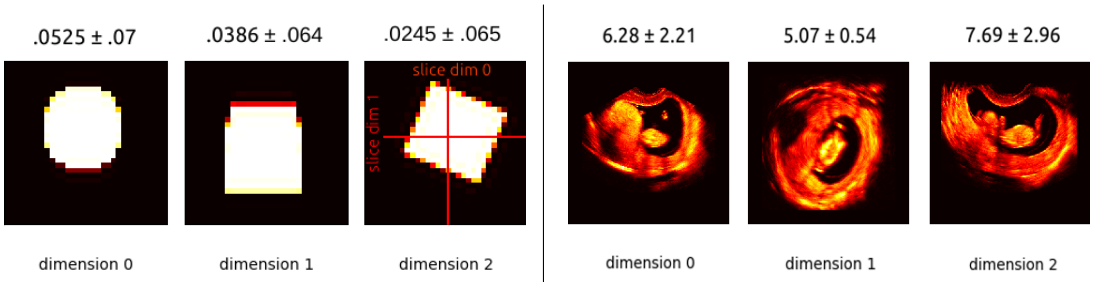
(a) Color-coded gestational age targets in latent space.

(b) Linear regression mean-squared errors in latent space.

Figure 4.4.: Visualization of gestational age targets and linear regression errors using a two-dimensional PCA projection in 32-dimensional latent space.

scored 1.0213. Standard deviations for both, respectively, were 0.013 and 0.036. The conventional autoencoder is slightly outperforming its slicing counterpart. However, variances do not allow for a statistically significant conclusion. Especially in the slicing setting variance is high. This is rather unexpected, as one would suggest that the smaller parameter space does not allow for large differences in latent spaces produced during reruns. Both scores are significantly better with the addition of an overarching autoencoding architecture with respect to the previous methods.

4.4. Slicing dimension



(a) Middle slices of an example cylinder. Above them are a comparison of MSE classification scores, experiments were rerun 20 times.

(b) Middle slices of an example ultrasound scan. Above them are a comparison of MSE regression scores, experiments were rerun 5 times.

Figure 4.5.: Middle slices across all three slicing dimensions.

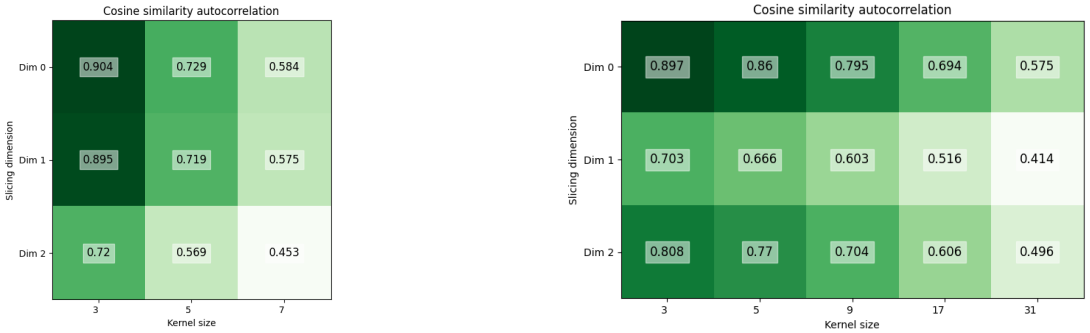
In figure 4.5, we can see the different MSE means and standard deviations for both datasets sliced across all three spatial dimensions. Table 3, shows the confusion matrices for geometric shape classification for different slicing dimensions. Classification should be read as the shape that is present in the volume with highest probability, according to the network.

For the geometry scans variances are quite similar but the third slicing dimension performs better than the first two. This may be the case because the rotational heterogeneity introduced in the dataset is in the plane aligning with the slices. This can also be observed from the middle slice examples in figure 4.5a.

In table 3 we can see the classification probabilities for all pairs of shapes, figure 4.7 visually lays out the same data using pie charts. First of all, it is interesting that the matrices are quite different for each slicing dimension. This may be the case because the projection of shapes onto the slices have become ambiguous depending on the slicing dimension. It seems tetrahedrons are causing most confusion for the other shapes, except when slicing in dimension 1. Spheres and cylinders have a high probability of being confused and cubes were often misclassified as cylinders in dimension 1, the middle slice examples show how the flat faces of rotated cylinders can look like cubes in certain slices.

What immediately stands out from the results in figure 4.5b is that both MSE variance and mean are significantly higher for the first and third slicing dimension in case of the ultrasound scans. Slicing across the cone (dimension 1) produced by the ultrasound machine produces the best results. In the middle slices of a random ultrasound scan in figure 4.5b, dimension 0 and 2 are shown to be ambiguous under rotational heterogeneity. Therefore it at first may not seem surprising their results both have similar, and higher, mean and variance. Intuitively, correlations are highest and position of the embryo in the slices is less ambiguous in dimension 1.

Figure 4.6 shows auto-correlations for all slicing dimensions and a range of kernel sizes, measured using cosine similarity. The algorithm described in subsection 2.4 was employed. In the geometry dataset, the kernel size was limited because of the smaller resolution. For both datasets and all slicing dimensions, correlations decrease as the kernel size increases. This is to be expected as the spatial distance between the slices translates to distance in feature space as well. Interestingly, the slicing dimensions that performed best in our previous experiment correspond to the dimensions with least correlation. The relation between slicing dimension performances does not change for different kernel sizes.



(a) Auto-correlations in the geometry dataset.

(b) Auto-correlations in the ultrasound dataset.

Figure 4.6.: Auto-correlations for varying kernel size and slicing dimension.

Confusion matrix comparison					
Dimension 0		cube	sphere	cylinder	tetrahedron
	cube	0.6875	0	0.2625	0.05
	sphere	0.0188	0.8625	0.05	0.0688
	cylinder	0.0312	0.0375	0.8625	0.0688
	tetrahedron	0	0	0	1
Dimension 1		cube	sphere	cylinder	tetrahedron
	cube	0.7437	0.05	0.2062	0
	sphere	0	0.975	0.025	0
	cylinder	0.025	0.05	0.925	0
	tetrahedron	0	0.0688	0	0.9312
Dimension 2		cube	sphere	cylinder	tetrahedron
	cube	0.9250	0	0.0063	0.0688
	sphere	0	0.9	0.075	0.025
	cylinder	0	0	0.95	0.05
	tetrahedron	0	0	0	1

Table 3.: Comparison of confusion matrices for different slicing dimensions. Consists of probabilities of shapes in rows to be classified as shapes in columns. Results are means over 20 runs.

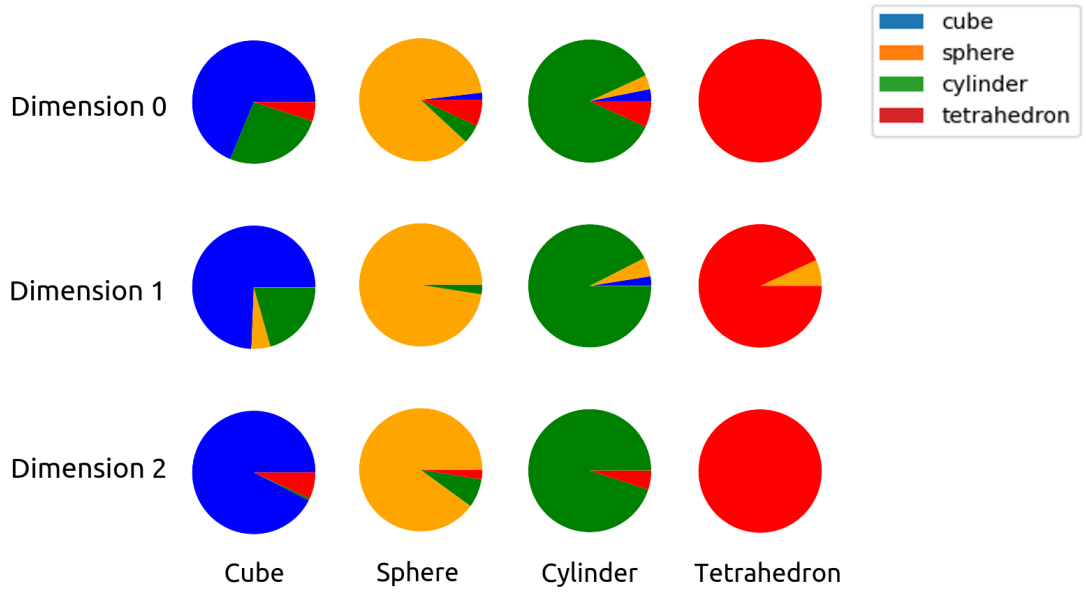


Figure 4.7.: Visualization of probability distributions of classification, given a shape in a volume sliced in a certain dimension.

5. Conclusion

In this report we have considered the slicing architecture for volumetric data, in the artificial and medical domains. They are aimed at resolving memory constraints, by breaking up dependencies in the data volumes across one of the spatial dimensions. Additionally, they are parallelizable and less complex in terms of parameters. The question this thesis addresses is whether these computational benefits weigh up to the compromise on expressibility.

The gathered results cover three scenarios; small noiseless volumes, large noisy volumes limited in number and autoencoding of the same noisy volumes.

In the initial scenario results had rather high variance but showed a significant performance increase when using the slicing architecture, despite saving on computation. In addition to that, the model practically does not overfit, while the conventional version dramatically does so. A noiseless context has limited stochasticity in the batching process, which is probably why slicing does not improve variance of results in these contexts.

Slicing the medical ultrasound volumes produced slightly different results. Final test set scores were not significantly different. This gives us an example of a context were employing a slicer does not significantly reduce performance and can be used to reduce resource requirements. The argument of generalization we have discussed, does not seem to hold in this context, both networks overfit to the same (considerable) extent.

Finally, we have turned to autoencoding architectures to investigate the need for additional regularization. Interestingly, the additional generalization of autoencoding seems to reduce the generalizing effect of slicing. In fact, the expressibility of the 3D convolutional method now converges to more similar solutions. Apparently the 3D convolutional model needs generalization to exploit the additional expressibility, which may be the reason why it is not performing better in previous experiments. The mean performances of the linear regression model in the latent spaces are not notably different between strategies, but they are significantly better than the non-autoencoders.

Our experimental approach indicated that the choice for a slicing dimension can cause large differences in results when it comes to performance and types of misclassifications. The results for the optimal dimension however, proved to be the opposite of our initial hypothesis. At least in the case of the two contexts considered, the similarity between slices across a slicing dimension is negatively correlated with the performance of the slicing model. It may be the case that the slicing models are still prone to overfitting, and prefer the subnetwork that encodes the most information to be the least complex. In other words, a slicing model that slices across the dimension with the most variance, will have to encode this variance with a smaller amount of parameters making it less prone to overfitting.

In conclusion, the usability of slicers are dependent on a number of factors. If the context does not require explicit generalization, or if slicers can be combined with other methods that have generalizing properties, 3D convolution may provide more expressibility. However, if little data is available, this expressibility may not be exploitable.

Slicing can make large models feasible in terms of computation when memory is limited. Acceptable sizes of batches with large input spaces may may mean such a model is needed.

Additionally, systems that have a design requirement of being small in size can use parallelizability to make specialised hardware that makes for additional speedup and reduction in memory requirements.

5.1. Limitations

In this section we will touch on limitations of the methods used and directions for possible future investigation of slicing methods for volumetric data.

Ultrasound data

Firstly, lets elaborate on the experiments with ultrasound data. The data considered is what we call raw, that is to say no (human) pre-processing techniques have been used to try to reduce the noise in the data. There are effective techniques that could be looked at, possibly increasing performance of the models or changing the relation between performances of the models. The same critique is valid for segmentation techniques, if the embryo could be succesfully segmented, this would significantly reduce input dimensionality and benefit model performance.

Only gestational age targets are used to test model performance. Other targets are available and could have been considered. Examples include birth weight, maternal pregnancy behaviour and miscarriages. They have not been considered in this research because of sparsity and class imbalance, additional challenges in the already challenging domain of noise and input space complexity.

Another limitation of our research is the reproducibility of the ultrasound results. The data used was provided by the Erasmus MC and is not publicly available. The models are available however, and can be evaluated in other domains with similar input size.

Additionally, the size of the dataset was very limited in the context of such a large input space. In the future, the Rotterdam Cohort programme will produce new batches of datapoints, which could be included to train more complex models that require additional data to be prevented from overfitting. These batches still vary in the technologies they were captured on (resolution) and newer ones contain a temporal dimension.

Choices for slicing

The slicing approach comes with two choices that have not been extensively discussed. Which slicing dimension is analytically optimal and what procedure is used in the recombining of slices.

Section 2.4 presents a method for choosing a slicing dimension. The approach considers correlations across such a dimension to create a measure for the amount of information that has to be encoded in the combination stage. This measure, however, is heuristic. It contains a hyperparameter k, indicating the size of the band of slices for which auto-correlations are calculated. Taking this k to be equal to the kernel size of compared 3D strategies may create a fairly robust algorithm for finding the slicing dimension that will compare best with respect to 3D convolutional counterparts. But choosing the best slicing dimension for standalone slicing methods requires a more thorough analytical approach. All correlations along the slicing

dimension should ideally be considered, at least per volume. This problem unfortunately is quadratic in slicing dimension resolution and quickly becomes intractable.

In our architecture, combination is done by a single layer with a linear learnable kernel across the slicing dimension. More research could be done to investigate the impact of (non-linear) multi-layer combination, using MLPs or convolution. Additionally, our model assumes a predetermined resolution for the slicing dimension to set as the combination kernel size. Temporal models could be looked at to target domains with variable resolution in one of the spatial dimensions. Computed Tomography scans are an example of a domain where specifically one of the three dimensions does not have constant resolution, making them unfit for convolution in this dimension if pooling is undesired. An LSTM or simple RNN could be used to combine slices into a single vector, by interpreting the variably sized spatial dimension as a temporal one.

Goals

Finally, let us discuss the main reason for using slicing methods. Apart from generalization, all considerations are related to computational cost. Is it justifiable to possibly compromise on interpretability and performance of models in critical domains like medical, when computation is so accesible in todays world [17][11]? Perhaps critical domains do not leave room for such additional risk. In the case of performance however, this consideration could always be made after evaluation of solution quality. For less critical domains, the argument could be made that computational cost is not just about economic cost but also about environmental cost. The artificial intelligence industry has a significant carbon footprint [22][19], caused by training of models in large clusters and distributed evaluation of trained models. Simpler slicing models may reduce this footprint while achieving similar performance.

A. Architectures

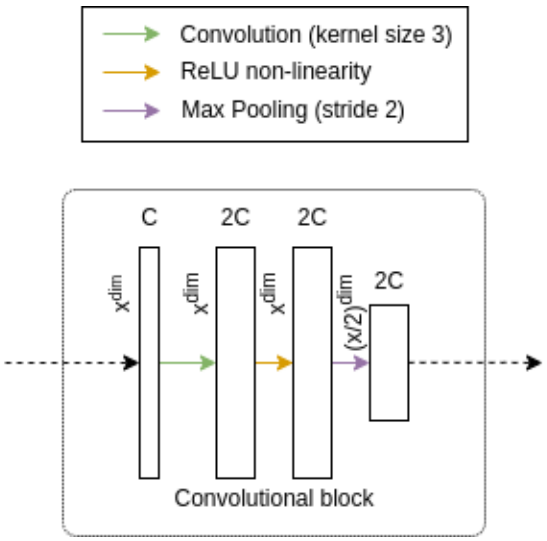


Figure A.1.: Abstract visualization of a single pass through a convolutional block for small input space.

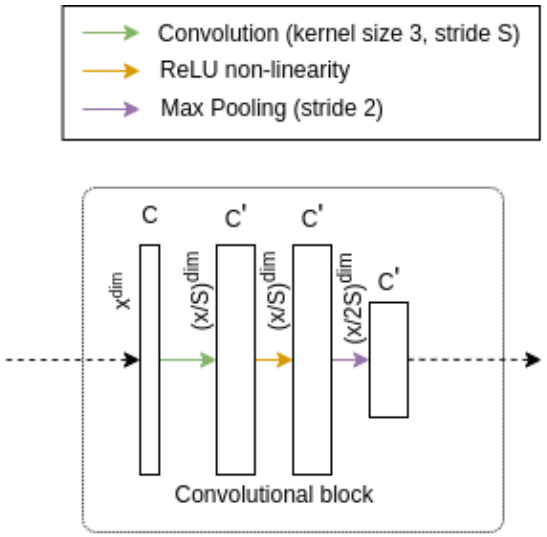


Figure A.2.: Abstract visualization of a single pass through a convolutional block for large input space.

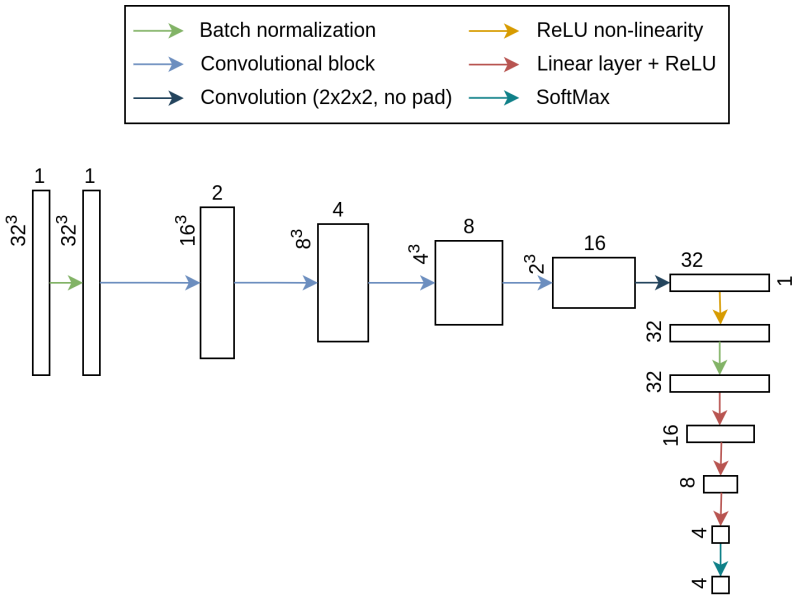


Figure A.3.: Visualization of the three-dimensional architecture for the 32^3 dimensional input space.

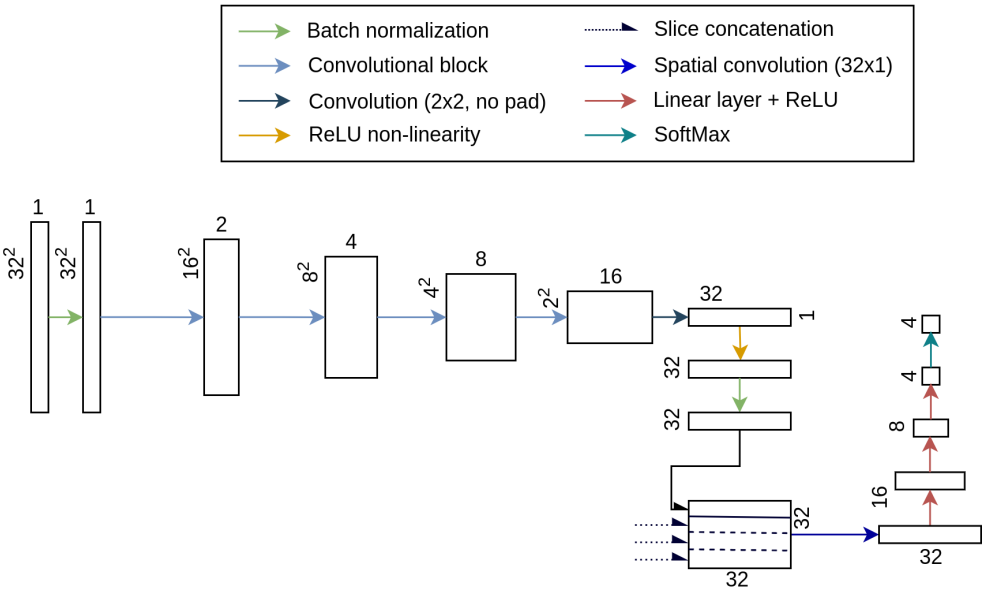


Figure A.4.: Visualization of the slicing architecture for the 32^3 dimensional input space.

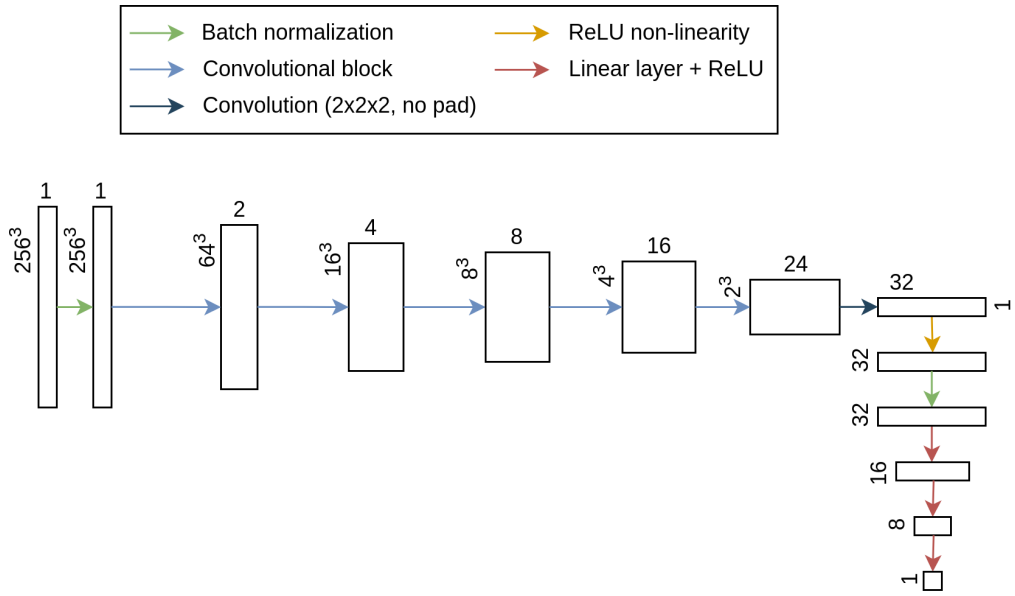


Figure A.5.: Visualization of the three-dimensional architecture for the 256^3 dimensional input space.

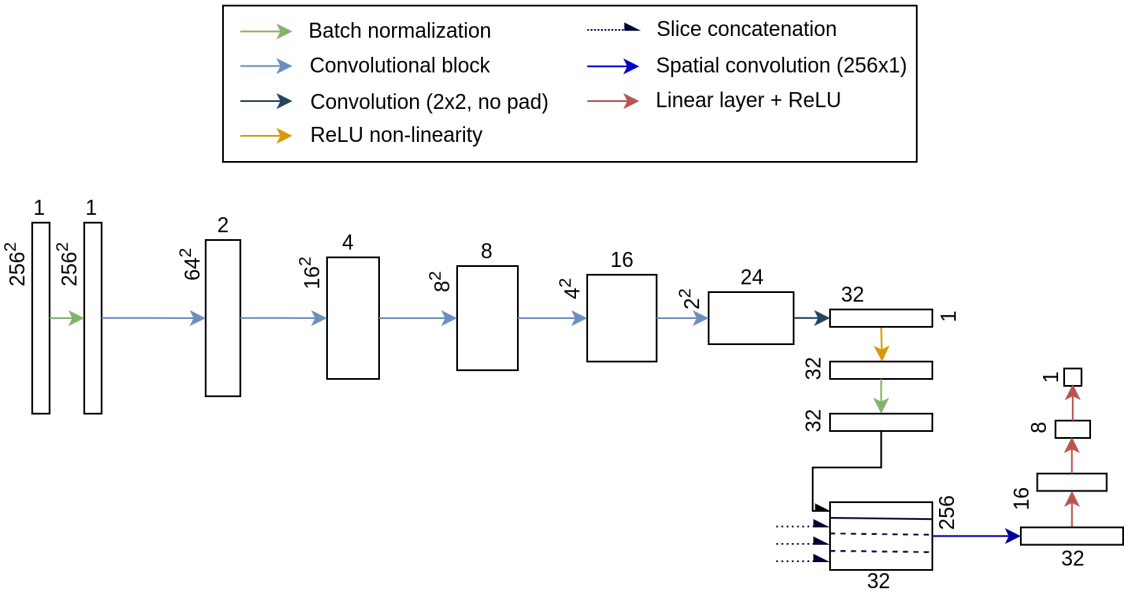


Figure A.6.: Visualization of the slicing architecture for the 256^3 dimensional input space.

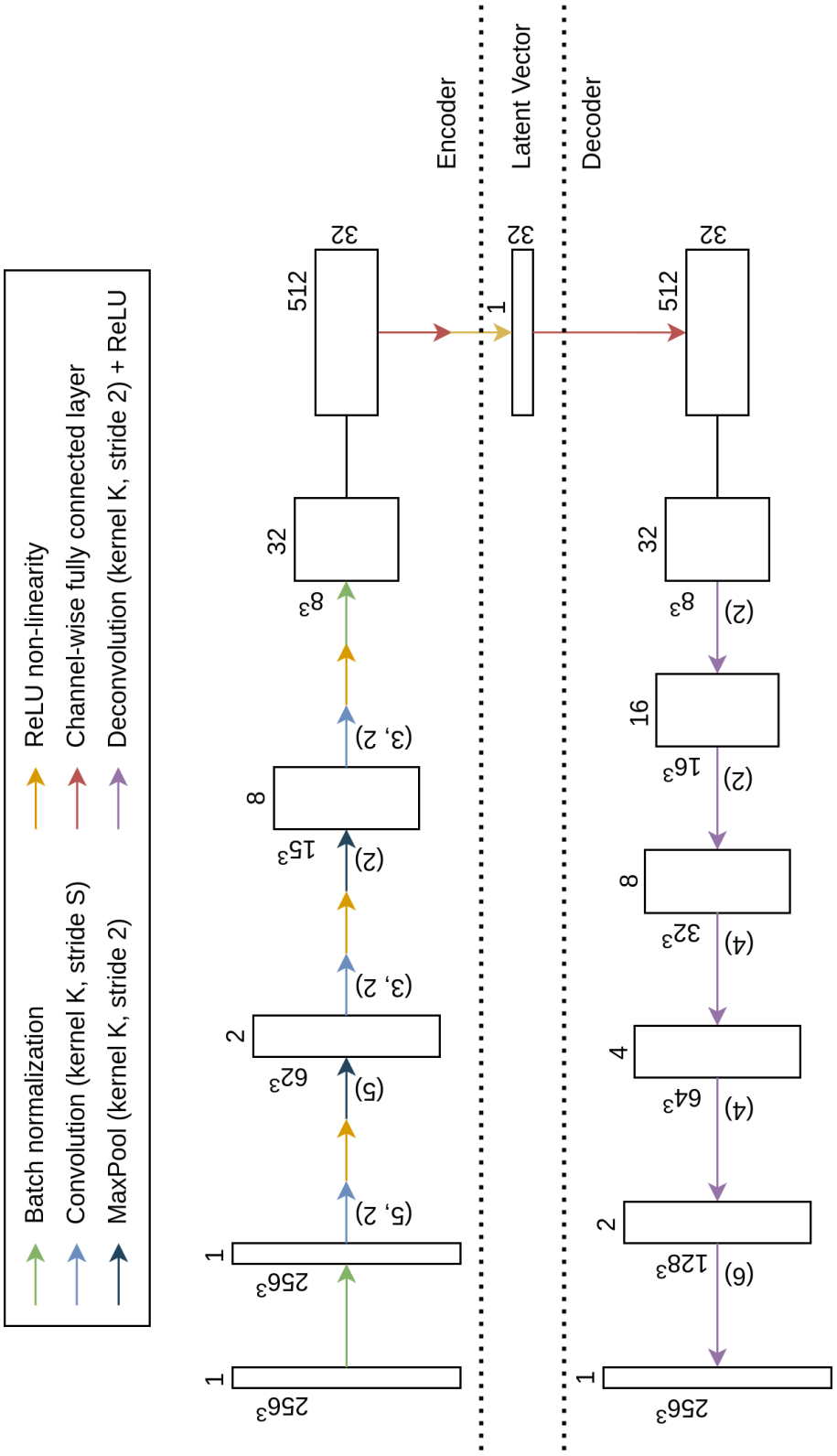


Figure A.7.: Visualization of the three-dimensional architecture for the ultrasound autoencoder.

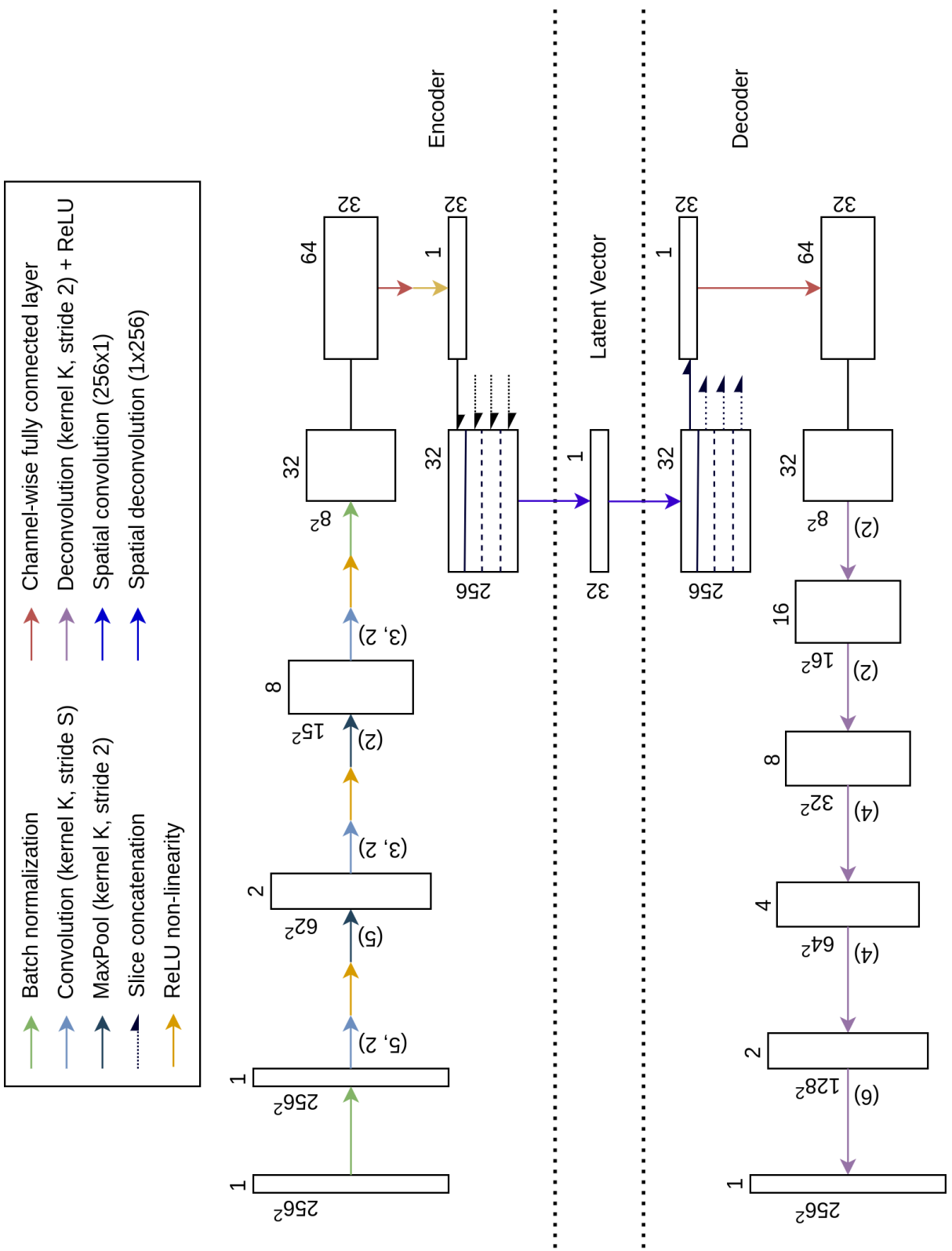


Figure A.8.: Visualization of the slicing architecture for the ultrasound autoencoder.

B. Definitions

B.1. Autoencoders

Autoencoders [18] are bilateral deep learning architectures. They are generally comprised of two subnetworks; an encoder and a decoder. The encoder is responsible for compressing the input space to a latent space, in computer vision this is usually done with convolution and multi-layered perceptrons. This compression is in turn fed to the decoding subnetwork, which tries to mimic the initial input with deconvolution. The layer that separates the encoder and decoder, is what we call the bottleneck layer. It contains less neurons than all other layers in the network, and its output are the vectors that live in the latent space.

The concept of autoencoding tries to tackle some issues that come with volumetric data. First of all, autoencoders are inherently regularizing, since they are forced to reduce the amount of information contained in the bottleneck layer. This eliminates some of the noise, as it can not be encoded in the latent space if it is not part of an underlying distribution of perfect observations. The lower dimensionality of the latent space with respect to huge input spaces will also reduce overfitting in downstream models.

Secondly, if input variance is high due to transformations like translation or rotation, the encoder may be forced to encode these variables in the latent space. If the encoder succeeds in doing this, the variance can be isolated in a number of single dimensions.

Because the network takes the input as its own target, we say it is self-supervised. If volumes are available but collection of corresponding targets is expensive, autoencoders can be trained on the images without targets. Then the encodings can be used in the training on a downstream target that is limited in number. It should be noted that taking this approach requires the autoencoder to be retrained and re-evaluated as new input or target data comes in; if the downstream network is retrained with newly acquired samples using the old encoder it will be biased towards samples the encoder has already seen.

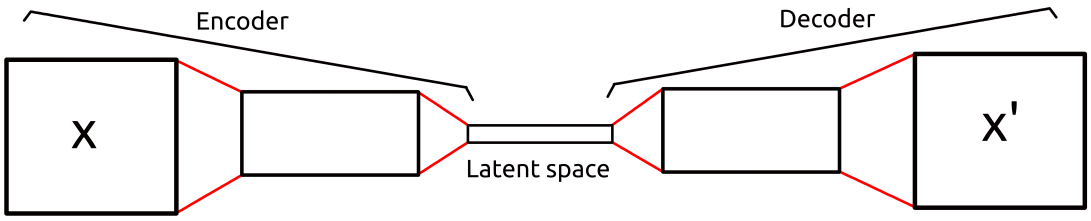


Figure B.1.: Schematic of the autoencoder architecture.

B.2. Parallelization

One of the discussed benefits of slicing is parallelization. In this section we will present a simplified mathematical definition of slice parallelization. Forwarding input through the network allows for parallelization in the reduction stage as well, but this follows from the function definition of slicing. Because this function definition follows the same principle that makes backpropagation parallelizable, we will restrict the scope of this section to backpropagation of output to the parameters in the slice reduction subnetwork.

An abstract mathematical representation of the slicing network can be seen in equation B.1, consult figure 1.1a for a visualization of the different subnetworks.

$$reduction \rightarrow R_{\theta_{red}(X)}, \quad combination \rightarrow C_{\theta_{com}(X)}, \quad downstream \rightarrow D_{\theta_{dow}(X)}$$

$$Y = D_{\theta_{dow}}(C_{\theta_{com}}(R_{\theta_{red}}(X))) \tag{B.1}$$

If we want to know what neurons in the reduction subnetwork are responsible in what way for the loss in some training iteration, we are interested in the derivative in equation B.2.

$$\frac{\partial loss}{\partial \theta_{red}} = \frac{\partial MSE}{\partial D_{\theta_{dow}}} * \frac{\partial D_{\theta_{dow}}}{\partial C_{\theta_{com}}} * \frac{\partial C_{\theta_{com}}}{\partial R_{\theta_{red}}} * \frac{\partial R_{\theta_{red}}}{\partial \theta_{red}} \tag{B.2}$$

The key in these equations is the combination function, which is not present in the 3D convolutional architecture. Equation B.3 is a description of this function and equation B.4 shows the derivative with respect to the reduction parameters, i indexes the position of the slice in the slicing dimension.

$$C(reductions) = \sum_{i}^{|slices|} \theta_{com_i} * R(slice_i) \tag{B.3}$$

$$\frac{\partial C_{\theta_{com}}}{\partial \theta_{red}} = \sum_{i}^{|slices|} \theta_{com_i} * \frac{\partial R(slice_i)}{\partial \theta_{red}} \tag{B.4}$$

The upstream derivative with respect to the reduction parameters is only dependent on the slice with index i, since reduction parameters are shared across the slicing dimension. This allows them to be computed in isolation and in parallel, before being combined. Assuming the combination and downstream subnetworks require asymptotically less memory resources than the reduction of a single slice, this reduces space complexity by a factor of $|slices|$.

Bibliography

[1] Eduardo Alcaín et al. “Hardware Architectures for Real-Time Medical Imaging”. In: *Electronics* 10.24 (2021), p. 3118.

[2] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[3] Pedro HB Diniz, Yi Yin, and Sally Collins. “Deep learning strategies for ultrasound in pregnancy”. In: *European Medical Journal. Reproductive health* 6.1 (2020), p. 73.

[4] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[6] Berend-Jan Lange. “Finding a rich lower dimensional representation”. In: (2021).

[7] Francesco Locatello et al. “Challenging common assumptions in the unsupervised learning of disentangled representations”. In: *international conference on machine learning*. PMLR. 2019, pp. 4114–4124.

[8] Pádraig Looney et al. “Automatic 3D ultrasound segmentation of the first trimester placenta using deep learning”. In: *2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017)*. IEEE. 2017, pp. 279–282.

[9] Francisco J Martinez-Murcia et al. “Studying the manifold structure of Alzheimer’s disease: a deep learning approach using convolutional autoencoders”. In: *IEEE journal of biomedical and health informatics* 24.1 (2019), pp. 17–26.

[10] J Alison Noble and Djamal Boukerroui. “Ultrasound image segmentation: a survey”. In: *IEEE Transactions on medical imaging* 25.8 (2006), pp. 987–1010.

[11] William D Nordhaus. “The progress of computing”. In: *Available at SSRN 285168* (2001).

[12] Kanghan Oh et al. “Classification of schizophrenia and normal controls using 3D convolutional neural network and outcome visualization”. In: *Schizophrenia research* 212 (2019), pp. 186–195.

[13] Deepak Pathak et al. “Context encoders: Feature learning by inpainting”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[14] Melek Rousian et al. “Cohort Profile Update: The Rotterdam Periconceptional Cohort and embryonic and fetal measurements using 3D ultrasound and virtual reality techniques”. In: *International journal of epidemiology* 50.5 (2021), pp. 1426–1427l.

[15] Sertan Serte and Hasan Demirel. “Deep learning for diagnosis of COVID-19 using 3D CT scans”. In: *Computers in biology and medicine* 132 (2021), p. 104306.

[16] Dinggang Shen, Guorong Wu, and Heung-Il Suk. “Deep learning in medical image analysis”. In: *Annual review of biomedical engineering* 19 (2017), p. 221.

[17] Benn Steil, David G Victor, and Richard R Nelson. *Technological innovation and economic performance*. Princeton University Press, 2002.

[18] Michael Tschannen, Olivier Bachem, and Mario Lucic. "Recent advances in autoencoder-based representation learning". In: *arXiv preprint arXiv:1812.05069* (2018).

[19] Aimee Van Wynsberghe. "Sustainable AI: AI for sustainability and the sustainability of AI". In: *AI and Ethics* 1.3 (2021), pp. 213–218.

[20] Anna Volokitin et al. "Modelling the distribution of 3D brain MRI using a 2D slice VAE". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 657–666.

[21] Maurice Weiler et al. "3d steerable cnns: Learning rotationally equivariant features in volumetric data". In: *Advances in Neural Information Processing Systems* 31 (2018).

[22] Carole-Jean Wu et al. "Sustainable ai: Environmental implications, challenges and opportunities". In: *Proceedings of Machine Learning and Systems* 4 (2022), pp. 795–813.

[23] Hiroyuki Yamaguchi et al. "Three-dimensional convolutional autoencoder extracts features of structural brain images with a "diagnostic label-free" approach: application to Schizophrenia Datasets". In: *Frontiers in Neuroscience* 15 (2021).