Institute for Natural Language Processing
University of Stuttgart
Pfaffenwaldring 5 b
D-70569 Stuttgart

Masterarbeit

Generation-Based Continual Learning Approach for Visual Question Answering

Pascal Tilli

Course of Study:	Software Engineering
Examiner:	Prof. Dr. Ngoc Thang Vu
Supervisor:	Dirk Väth, M.Sc.
Commenced:	February 2, 2020
Completed:	July 27, 2020

Abstract

Humans have the ability to continually acquire knowledge throughout their lifespan. In contrast, neural networks suffer from catastrophic forgetting when trained on new tasks. Continual learning studies the methods to achieve similar memory effects in artificial neural networks and enable them to learn tasks sequentially. In this thesis, we investigate generation-based continual learning methods. Generation-based models have been used to replay previously learned data distributions and retain knowledge of solving previous tasks. Generative replay has been shown to work on uni-modal datasets with relatively low complexity. Our experiments focus on Visual Question Answering (VQA), which is known to be a more complex, multi-modal domain. We provide approaches and results for three datasets of the domain VQA and one uni-modal toy dataset. As a proof of concept, we start by training the handwritten digits of the MNIST dataset continually. For the VQA domain, we study the VQAv2 dataset, CLEVR, and Shapeworld. We found that generative models do not perform well in VQA. Our models could not overcome catastrophic forgetting except for the Shapeworld dataset. Within the Shapeworld setting, our approach with generative replay did enable continual learning.

Zusammenfassung

Menschen besitzen die Fähigkeit sich kontinuierlich, im Laufe ihres Lebens, Wissen anzueignen. Neuronale Netze hingegen leiden unter dem Phänomen des verhängnisvollen Vergessens, wenn es hinsichtlich neuer Aufgaben trainiert wird. Das Feld des kontinuierlichen Lernens befasst sich mit den Methoden, um vergleichbare Erinnerungseffekte in künstlichen neuronalen Netzen zu erzielen, um sie in die Lage zu versetzen, Aufgaben sequentiell zu lernen. In dieser Thesis untersuchen wir generierungsbasierte Methoden des kontinuierlichen Lernens. Generierungsbasierte Methoden werden verwendet, um zuvor gelernte Datenverteilungen wiederzugeben, um das Wissen zur Lösung früherer Aufgaben zu erhalten. Es hat sich gezeigt, dass die generierte Wiedergabe von Datenpunkten zur Erhaltung des Wissens bei unimodalen Datensätzen mit relativ geringer Komplexität funktioniert hat. In unseren Experimenten fokussieren wir uns auf Visual Question Answering (VQA), eine bekanntlich komplexe, multimodale Domäne. Wir stellen Ansätze und Ergebnisse für drei Datensätze der Domäne VQA, sowie für einen unimodalen Datensatz zur Verfügung. Als Proof-of-Concept beginnen wir damit, die handschriftlichen Ziffern des MNIST-Datensatzes kontinuierlich zu trainieren. In der Domäne VQA untersuchen wir den VQAv2-Datensatz, CLEVR und Shapeworld. Die Ergebnisse zeigen, dass generative Modelle in VQA nicht gut funktionieren. Unsere neuronalen Netze konnten das katastrophale Vergessen, mit Ausnahme des Shapeworld-Datensatzes, nicht überwinden. Mit dem Shapeworld-Datensatz konnte unser Ansatz der generativen Wiedergabe kontinuierliches Lernen ermöglichen.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AI Artificial Intelligence. 17

CL Continual Learning. 18

CLS Complementary Learning Systems. 17

CNN Convolutional Neural Network. 24

CV Computer Vision. 22

DEN Dynamically Expanding Network. 34

DL Deep Learning. 23

EM Earth-Mover. 35

EWC Elastic-Weight-Consolidation. 34

GAN Generative Adversarial Network. 17

GP Gradient-Penalty. 30

JS Jensen-Shannon. 35

KL Kullback-Leibler. 35

LSTM Long-Short-Term-Memory. 24

LWF Learning Without Forgetting. 34

ML Machine Learning. 17

MLP Multilayer Perceptron. 24

NLP Natural Language Processing. 22

PCA Principal Component Analysis. 30

ReLU Rectified Linear Unit. 25

RNN Recurrent Neural Network. 26

sigmoid Logistic Function. 25

SVD Singular Value Decomposition. 31

SVHN Street View House Number. 34

tanh Hyperbolic Tangent. 25

TV Total Variation. 35

VQA Visual Question Answering. 3

WGAN Wasserstein GAN. 18

WGAN-GP Wasserstein GAN with Gradient-Penalty. 9

1 Introduction

Humans and animals have the ability to continually acquire knowledge and skills throughout their lifespan. These skills can be fine-tuned. Previously learned knowledge can be transferred to acquire new knowledge. This ability is called continual or lifelong learning and is obtained by complex neurocognitive mechanics that enable the human long-term memory consolidation and retrieval. Concluding from this, it is necessary for computational systems and autonomous agents to get abilities like lifelong learning to successfully interact with the real world and process continuous streams of information. A system needs to learn multiple tasks from dynamic data distributions. [PKP+19]

The capability to continually learn new tasks over time by retaining the information of previously learned tasks is referred to as continual learning. This has been, and still remains a huge challenge for Machine Learning (ML) and neural networks, in order to develop an Artificial Intelligence (AI). One of the main issues of neural networks is their weakness of catastrophic forgetting. Learning new tasks with new information (data) interferes with previously learned tasks. [MMO95] This leads to a phenomenon that the performance on previously learned tasks decreases drastically and the old information is completely overwritten by the newly learned knowledge. In order to adapt changes in the data distribution, the parameters of a neural network need to be trained from scratch, assuming all training data of all tasks that need to be learned are available and there is sufficient memory. Practically spoken, this approach is very inefficient, if new data needs to be learned in real time. [PKP+19]

We want to investigate if continual learning can be applied in a complex domain like VQA, which combines different areas like vision and language. A VQA system takes an image and a free-form natural language question about the image as input and produces a natural language answer as output. Answering a question about an image requires the ability to identify specific properties in the image with respect to the given question. This could be a color or a shape of an object as well as an activity of a person. No limitations should be set to avoid biases in order to create a VQA system that is able to answer arbitrary questions about any image.

To enable continual learning in VQA systems, we need to evaluate different continual learning methods. We focus on the Complementary Learning Systems (CLS) theory that illustrates the importance of dual memory systems including the hippocampus and neocortex. The hippocampus processes recent experiences, which get reactivated during sleep or conscious and unconscious recalls [GMH+08]. The memory gets consolidated in the neocortex through the activation of the encoded experiences [OPDC10]. Our approach is similar to the approach of Shin et al. [SLKK17]: we use generated pseudo-data to replay previously acquired knowledge, known as generative replay. In particular, we use a Generative Adversarial Network (GAN) to generate these data samples. The data is then paired with the data from the current task that the model should learn. Afterwards, the generator gets trained on the data of the current task as well as on generated data from the previous ones to produce data samples from all tasks seen so far.

GANs are an important part in the context of generative replay, because the knowledge can only be retained if the generated samples are of a comparable quality. They belong to the category of generative models, which are trained as a game between to neural networks. A generator network produces synthetic data given some noise vector and a discriminator network discriminates between the real data and the output of the generator. GANs can produce data samples of high quality, however, they are often hard to train, because they often fail to converge [GAA+17]. Arjovsky et al. [ACB17] proposed an alternative to the original GAN, called Wasserstein GAN (WGAN), which uses the Wasserstein distance to produce a value function that improves the results and training stability. We focus on an improved version, called WGAN-GP, which omits the weight clipping of the original WGAN. The results are of higher sample quality and the WGAN-GP fails less often to converge [GAA+17].

We experiment with multiple datasets of the domain VQA. Our objective is to compare the complexity of these datasets and evaluate the capability of generative replay to enable continual learning. Our results show the limitations of generative replay for each dataset and point out the reasons why our approach succeeds or fails. The properties of the datasets play an important role when we split the datasets into tasks to train them sequentially. We investigate another research question, which addresses the problem of feature vector generation instead of generating real-like data samples. For multi-modal problems, generating a combined feature vector avoids the difficulty of generating each input type individually. A problem of generating the different input channels individually is to verify if they have a relation to each other; there might be none. Generative replay relies on the sample quality of a generator. We want to investigate the capabilities of state-of-art GANs whether they can be applied on VQA datasets and capture the underlying data distribution. We derive the research question, if GANs can capture and reproduce the distribution of feature vectors.

Initially, this thesis focuses on the background information needed. We describe the CLS in more detail, followed by Continual Learning (CL) and VQA. We review related works of the different research areas to determine recent findings and limitations. The following section describes the used methods to address our problem, before we go into details about our approach and experimental results. The last parts of this thesis contain a discussion about our findings and give an outlook about future works in these areas.

2 Background

2.1 Complementary Learning Systems

This section covers biological aspects of continual learning as it occurs in humans and mammalians. These aspects are important, as the idea of this thesis relies on these key concepts and explains their origin.

Humans have the capability to adapt to various and very diverse challenges by acquiring new knowledge and skills, refining and reusing them across multiple domains [BLS12]. However, humans still tend to forget already learned information, but this rarely occurs while acquiring new knowledge [Fre99]. Taking a look at how our brain learns and memorizes, understanding these principles can help enabling the same effect in artificial neural networks. Different brain areas operate at multiple timescales and learning rates, and thereby differing significantly in a functional way [BF16].

The CLS is a prominent example, explaining the different parts of the human brain and their contribution towards learning and memory consolidation. In detail, it focuses on the neocortex and hippocampus and their contribution and interaction between each other with respect to the human memory capabilities. The hippocampal system represents the short-term memory and is able to adapt rapidly to new tasks or information, which will be played back to the neocortex over time, see Figure 2.1. The neocortical system describes the long-term memory, thus enabling humans to retain information over their lifespan. Consequently, the hippocampus has a rapid learning rate and encodes sparse representations of events to minimize interference. In contrast, the neocortex employs a slow learning rate and builds overlapping representations of the learned knowledge. The interaction between hippocampus and neocortex is crucial to generalize over tasks and learn regularities as well as specifics from episodic memories. [MMO95]

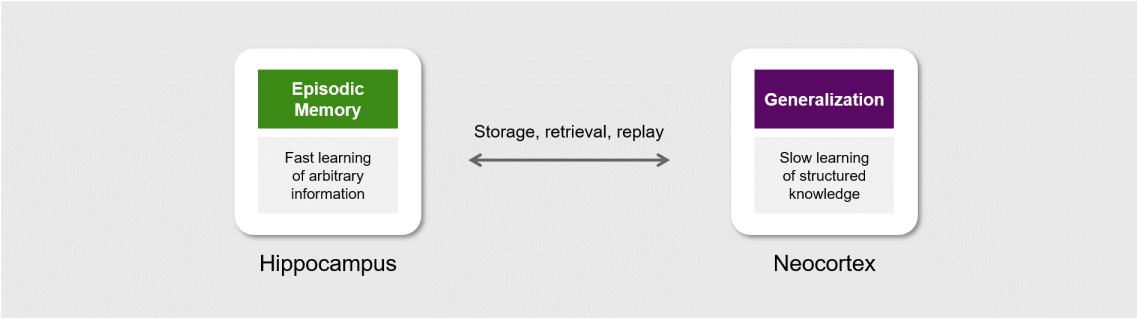

Figure 2.1: The complementary learning systems (CLS) theory [MMO95] comprising the hippocampus for the fast learning of episodic information and the neocortex for the slow learning of structured knowledge. Adapted from [PKP+19].

Recent findings suggest, that the role of replaying memories stored in the hippocampus is not only responsible for integrating new information, but also supports the goal-oriented manipulation of experience statistics [OPDC10]. The hippocampus is thereby responsible for creating episodic events that are activated during sleep or conscious and unconscious memory recall [GMH+08] in order to consolidated and generalize the information in the neocortex by reactivating these memories in form of internally generated replays [Rat90]. However, the hippocampus supports additional forms of generalization through the recurrent interaction of episodic memories [KM12]. Another finding suggests that, if the new information is consistent with existing knowledge, the transmission into the neocortex is significantly faster than, if the new information does not intersect with already known information [TTK+11]. All in all, the CLS states the effectiveness of the human brain in order to generalize across experiences while retaining specific memories in a lifelong manner. The exact neural mechanisms still remain poorly understood [PKP+19].

This thesis aims to transfer the theory of CLS into the field of AI. The idea is to enable CL by rebuilding neural networks that incur the functionality of the hippocampus and neocortex. The following chapters explain the used methods to achieve similar memory effects. We start by going into detail about continual learning, before we move on to our application, which is visual question answering, and finally we cover our methods belonging to the field of machine learning.

2.2 Continual Learning

This section is based on Awasthi and Sarawagi [AS19].

Continual learning refers to the principle of retaining and reusing previously learned knowledge to learn new tasks in shorter training time and with fewer resources. It is inspired by the lifelong learning capability of humans where tasks are being learned cumulatively instead of learning each task from scratch. Formally, tasks $T_1, T_2, ..., T_n$ with their respective datasets $D_1, D_2, ..., D_n$ are being trained sequentially, by training task T_i on data D_i, to eventually gain the capability to solve all tasks. Thereby the trained model needs to provide a high accuracy on all previous learned tasks, instead of just the most recent one. Training task T_i is expected to benefit from the previous trained tasks, i.e. the knowledge of the previous tasks is expected to speed up the training of task T_i. Also, a backflow of knowledge could improve solving previous learned tasks.

The following paragraphs list challenges of CL in neural networks.

Catastrophic Forgetting is a huge problem when it comes to learn tasks in a continuous manner. The backpropagation adapts the parameters to the most recent task, leading to a poor performance on previously learned tasks. Learning in neural networks is forgetful, making it impossible to train changing task objectives or changing data distributions.

In some situations, it may not be feasible to store and reuse training data from previously learned tasks, due to memory constraints.

Tasks may differ in semantics and syntax. On the one hand, tasks sharing similar syntax may differ semantically (e.g. text generation of news articles or text generation of poems), on the other hand tasks might differ in syntax of their input and output (e.g. sentence classification vs sentence generation).

This chapter is divided into three categories. The first category addresses CL on importance weighted parameter regularization, which makes learning in neural networks less forgetful. The second category contains data-replay approaches that store samples from previously learned tasks or generate samples from previously seen data distributions. The third category is about methods which enable neural networks to grow over time while learning new tasks to accommodate new knowledge.

2.2.1 Regularization-Based Approaches

Catastrophic forgetting, which prevents neural networks to learn new tasks sequentially, can be overcome by regularizing the weights of neural network. A common method within these approaches is to identify the most important weights of previously learned tasks in order to protect them from further updates. Unimportant weights can still be updated to learn new tasks. Kirkpatrick et al. [KPR+17] proposed a solution, that calculates the importance factor of the weights by computing the fisher information for all weights. A downside is, that the parameter importance for a task is only computed after the task has been learned and does not consider the learning process itself. The effectiveness of regularization-based approaches is limited to a certain number of tasks by the finite capacity neural networks.

2.2.2 Dynamic Neural Networks

Approaches using dynamic neural networks try to overcome catastrophic forgetting by adapting the structure of the neural network. These methods begin with a simplified network architecture and extend their architecture incrementally with new components to achieve satisfactory performance on the current task [AS19]. The change of the architectural properties enables to learn each task separately, but although they perform well, in most cases the need for potentially multiple models for each new task results in a growing computational and memory requirements, which do not seem to truly solve continual learning [FG18].

2.2.3 Data-Replay-Based Approaches

Approaches within this group are divided into two sub-groups. The first kind of approaches assumes modest memory space to store samples from previously learned tasks, which can be used during the training of a new task to keep the performance on the old tasks as well as achieving to learn the new task. An alternative approach would be to learn the data distributions from previous tasks and draw pseudo samples during the training of a new task. Again, the idea is to keep the performance of previously learned tasks stable and on the other side, still learn a new task. Drawing pseudo samples from a learned data distribution falls within the scope of generative models.

This thesis focuses on generative replay approaches to achieve CL in VQA. Awasthi and Sarawagi [AS19] stated the question whether the generative replay approach can scale up to more complex domains, which we want to investigate by applying the approach in VQA. The next section focuses on different continual learning scenarios.

2.2.4 Continual Learning Scenarios

van de Ven and Tolias [VT19] proposed three continual learning scenarios for a more structured comparison. They distinguish between task-incremental learning, domain-incremental learning and class-incremental learning.

In the task-incremental learning scenario, models contain the information which task is being performed. It is thereby possible to train the neural networks with task-specific components, for example using a ”multi-headed” output layer that contains different output layers for each task. This is considered to be the easiest continual learning approach.

In the domain-incremental learning scenario, models are unaware which tasks is being performed. However, it is not necessary for the model to know which task it is performing, because the structure of the tasks stay the same, only the input distributions changes.

In the class-incremental learning scenario, the model needs to learn to solve all tasks seen so far and determine which task they are currently shown. This is considered to be the closet to real-world problems of incrementally learning new classes of objects.

We define our continual learning tasks according to the class-incremental learning scenario, because it represents real-world continual learning problems. In our opinion, this makes the most sense to address and potentially solve the actual problem.

2.3 Visual Question Answering

VQA combines Computer Vision (CV) and Natural Language Processing (NLP). The input of a VQA system is an image and an open-ended, natural-language question, without any form restrictions, about the image. The output is a natural language response. Open-ended questions require a set of capabilities to be answered by an AI. These capabilities are fine-grained recognition (e.g., "What kind of cheese is on the pizza"), object detection (e.g., "How many bikes are there?"), activity recognition (e.g., ”Is the man crying”), knowledge base reasoning (e.g., ”Is this a vegetarian pizza?”), and commonsense reasoning (e.g., ”Is this person expecting company?”). Many questions can be answered with ”yes” and ”no”, while on the other side many questions are far from being answered as trivial. Questions about images often tend to seek specific information, that can often be answered by one-to-three word answers. [AAL+15]

As a multi-modal problem, VQA has received more attention in the past years [AAL+15; GKS+17; ZGS+16]. Antol et al. [AAL+15] state that an ”AI-complete” task should require multi-modal knowledge beyond a single sub-domain and should have a quantitative evaluation metric to track the progress. Domains like CV and image captioning may not be as ”AI-complete” as desired [AAL+15]. As a consequence, Antol et al. [AAL+15] created a large dataset in the domain of VQA, containing over 200,000 images from the MS COCO dataset [LMB+14] and 760,000 questions with approximately 10 million answers. VQA offers a large set of challenges to solve, such as combining NLP and CV.

Goyal et al. [GKS+17] proposed an improved version of the previously mentioned dataset. Our models within this work are trained with the second version of the open VQA dataset (VQAv2), which tends to be more balanced compared to the first version [GKS+17]. Analysis of the previous

dataset with respect to their models led to the conclusion that the models could reach superficially good performance, without truly understanding the visual content of the images. For example, the questions starting with ”What sport is...”, in 41 % of the cases have the answer ”tennis” and the questions beginning with ”How many...” can be answered correctly by saying ”2” with an accuracy of 39%. Therefore, the model does not explicitly need to learn to how to interpret and understand images, rather than just learning the answer distributions of each question type, and ignoring the information stored within the images. Inside the balanced dataset, each question is linked to complementary images, so the answer of the questions results in different solutions. Furthermore, all models trained on the first version of the VQA dataset result in a significantly lower accuracy when being trained on the second, balanced VQA dataset [GKS+17]. The hypothesis of the Goyal et al. [GKS+17] is that if each question Q has two different answers, A and A', for two different images, I and I', the only way to learn how to answer the question correctly is by looking at the image. Therefore, the idea is to make it impossible for language and vision models to exploit language priors, and to focus for more on the image understanding part. The dataset contains of approximately 1.1 Million (image, question) pairs and approximately 13 Million associated answers [GKS+17].

2.4 Machine Learning

2.4.1 Deep Learning

Deep Learning (DL) represents a sub-group of ML, which describes models of multiple layers to learn representations of data with multiple levels of abstraction. These methods have drastically improved state-of-the-art speech recognition, visual object recognition, object detection and fields like drug discovery and genomics. DL discovers the hidden structure of large datasets by using back-propagation algorithms in order to adapt its parameters in order achieve the defined objective [LBH15].

DL methods are representation learning methods, which are in turn a set of methods, that allows a machine to be fed with raw data to automatically discover the representation needed for detection or classification. These methods contain multiple levels of representation with simple, but non-linear functions to obtain different representations at different levels, that are passed into the next layer, obtaining a slightly more abstract representation. At the beginning, the raw input is given into the network, passed through every layer until the output representation occurs at the last layer. With the composition of enough layers (such transformations), complex functions can be learned. Given a classification task, the higher levels of representation amplify the relevant information and suppress not relevant information, within an image for example, to get the correct result. An important part of these methods is, that these representation layers (that learn the features) are not designed by a human engineer, but instead learned by the system [LBH15].

To train a DL model a large dataset is needed, containing as many samples as possible. For a classification task of images, this could be pictures of different animals, which the classifier should be able to distinguish. During training, the network is given an image and produces an output, in this example one for each class (animal). The output is a vector of scores, in which the desired category should have the highest value (score) of all categories. Before the network is trained, the output vector is probably going to produce arbitrary results for each class. Therefore, an objective

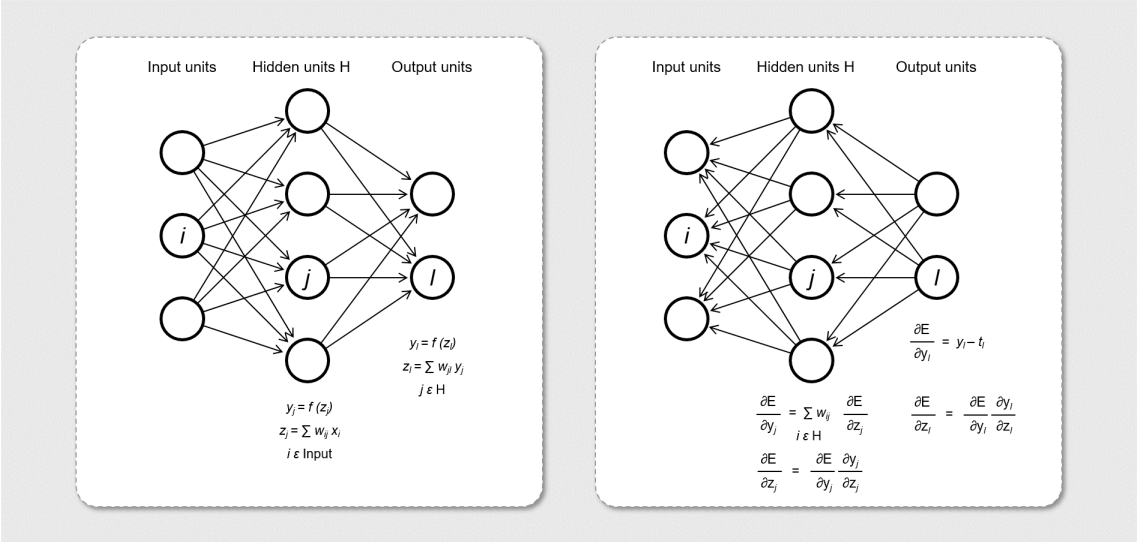

Figure 2.2: Multilayer Perceptron (MLP) with one hidden layer. Forward- and backward-pass are displayed. Adapted from LeCun et al. [LBH15].

function needs to be defined, to measure the error between the produced output and the desired output. Using the error of each data sample, the internal parameters (weights) can be updated in order to reduce the error. To properly update these parameters, which are real numbers, the learning algorithm computes a gradient vector for each weight, that indicates how to the error would increase or decrease, if the weight was increased by a tiny amount. The weight vector is then updated in the opposite direction of the gradient vector. The negative gradient vector indicates the direction of updates to get closer to a minimum, where the output error is low on average [LBH15].

The following sections are about the used methods within this work. First, one of the most basic neural networks, the Multilayer Perceptron (MLP) is shown. Furthermore, the Long-Short-Term-Memory (LSTM), which is good at learning long term dependencies and therefore successful in natural language processing, and the Convolutional Neural Network (CNN), which is good at learning high level features of objects within images and videos, are explained. Finally, the idea of a GAN is displayed. It learns the given data distribution in order to recreate it, i.e. generate new samples that match the data distribution and cannot be distinguished from the real data.

Multilayer Perceptron

A MLP is a neural network with multiple layers. It contains at least one hidden layer; it therefore has a minimum of three layers. Figure 2.2 displays a MLP with one hidden layer, a total of three layers. Apart from this, the left side of the graphic displays the forward-pass, the right side shows the backpropagation of an error (also called backward-pass). To compute the output of a neural network, the input needs to be passed through all layers, being transformed at each layer and fed into the next layer. At each layer, the first step is to compute z_i of each unit (node), which is the weighted sum of the outputs of the units of the layer below. Afterwards a non-linear function $f(.)$ is applied to z_i to obtain the output of a layer [LBH15].

$$f(z) = max(0, z) \tag{2.1}$$

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.2}$$

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

Non-linear functions, also known as activation functions, can be all different kinds of functions. The most common ones are Rectified Linear Unit (ReLU), shown in Equation (2.1), its variants and more conventional sigmoids, such as the Hyperbolic Tangent (tanh), shown in Equation (2.2), and Logistic Function (sigmoid), shown in Equation (2.3).

The right side of Figure 2.2 shows the back-propagation process. At each hidden layer the error derivative is computed with respect to the total inputs of the in the layer above. The error derivative with respect to the output gets converted to the error derivative with respect to the input by multiplying it by the gradient of $f(z)$. The error derivative with respect to the output at the output layer is computed by differentiating the cost function [LBH15].

Convolutional Neural Networks

CNNs have shown great success in applications like image classification, for example face or handwriting recognition [LHB04]. Convolutional networks are based on three core ideas: local receptive fields, shared weights and spatial or temporal sub-sampling. Neurons can extract elementary visual features such as edges, endpoints or corners with local receptive fields. The subsequent layers combine these features to extract higher level features. Shifts of the input can be useful, because feature detectors that are useful on one part of the image might be useful across the entire image [LBBH98]. This leads to the phenomenon that receptive fields located at different places on the image have identical weight vectors [RHW85]. Units (neurons or a single weight) in a layer are organized in feature maps and perform the same operation on different parts of the image. A convolutional layer consists of several feature maps in order to extract multiple features at each location of the image [LBBH98]. Figure 2.3 shows a CNN with multiple convolutional layers.

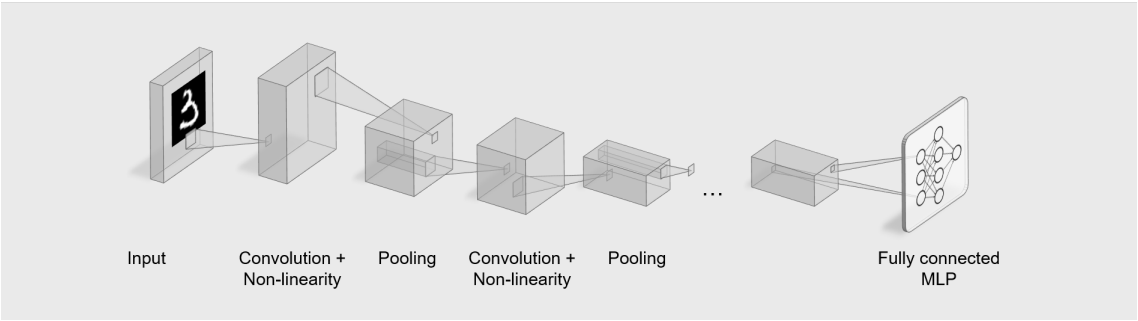

Figure 2.3: Convolutional Neural Network.

Pooling layer are used in-between convolutional layers. The idea is to reduce the number of parameters and the spatial size of the representation. It also reduces the computational time and controls overfitting. A pooling layer applies an operation, for example the *MAX* operation, using filters of a specified size. Figure 2.4 shows an example of a max pooling applied on a four times four matrix. The filter size is two times two with a stride of two, which leads to a reduction of 75%

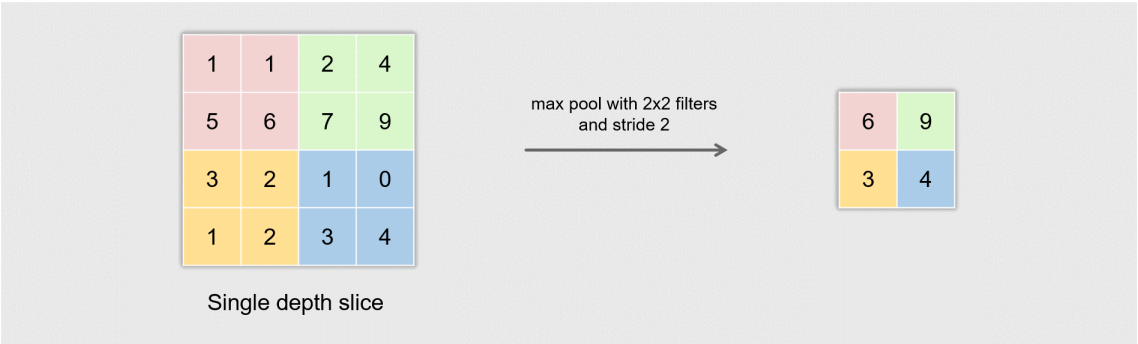

Figure 2.4: Example of max pooling.

of the original size of the matrix. The *MAX* operation extracts the highest value of the specified filter region. Pooling layers are used to downsample the size incrementally.

After convolutional layers, with or without max pooling layers, the output usually gets flattened and fed into a few fully-connected layers to finally get the result. Fully-connected layers are basically a MLP as it is described in Section 2.4.1.

Transposed Convolutional Neural Networks

Transposed convolution arises from the desire to use a transformation going into the opposite direction of a regular convolution, described in the previous Section 2.4.1. The goal is to get an output of the shape similar to the input of the regular CNN [DV16]. In the field of GANs, this is relevant for our generator and critic (discriminator). The critic learns to distinguish the real data distribution from the generated data distribution by applying regular convolution. The output is a scalar value. The generator reproduces the generated data by applying transformations in the opposite direction, i.e. transposed convolution. Transposed convolution is also called deconvolution or fractionally strided convolutions.

The difference whether it is a regular convolution or a transposed convolution is determined by the computation of the forward and backward passes. Convolution can be represented as a sparse matrix C. The backward pass can be obtained by transposing C. The error is backpropagated by multiplying the loss with C^T. Concluding, forward and backward passes are computed by multiplying the kernel with C and C^T respectively. The forward and backward passes of a transposed convolution are computed by multiplying with C^T and $(C^T)^T = C$ respectively [DV16].

Long-Short-Term-Memory

A LSTM is a specific kind of Recurrent Neural Network (RNN) to model temporal sequences and their long term dependencies more accurately than a conventional RNN. RNNs are a more powerful tool to model sequence data, like speech, than feed-forward neural networks such as MLPs, due to their cyclic connections [SSB14]. RNNs process their sequential input one element at a time, storing the information in a hidden state, which represents the information about all previous

inputs. Training of RNNs proved to be difficult, due to the back-propagation process, because the back-propagated gradients either shrink or grow at each time step, which leads to exploding or vanishing gradients [HS97].

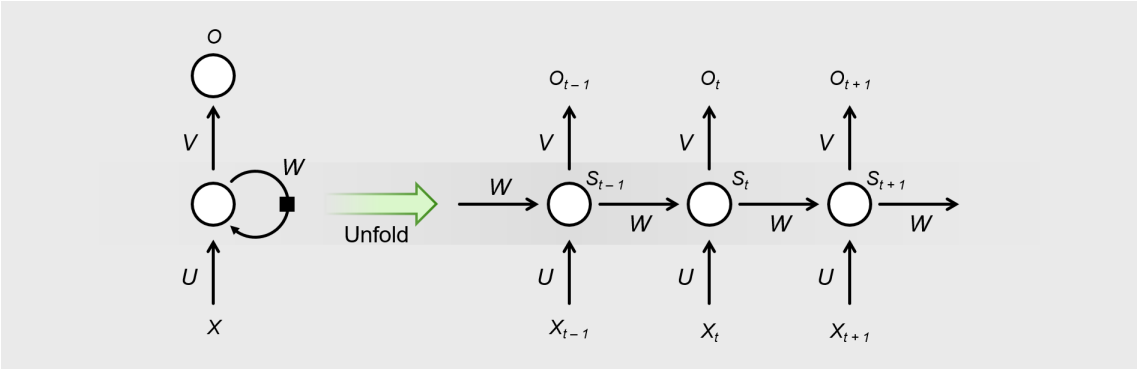

Figure 2.5: Recurrent Neural Network (RNN) and the unfolding in time. Adapted from LeCun et al. [LBH15].

Figure 2.5 shows the unfolding of a RNN in time. The left side shows the repetitive, sequential process of calculating the output o given an input x. The neuron s gets the input in a sequential order, merging the input at time step t with the information of the previous step, stored within the hidden state. The black square displays a delay in time, showing the passing through the neuron s repetitively. The right side shows the unfolding in time, how the output is computed at each time step t. The hidden state gets computed by weighting the input x_t with matrix U and adding the weighted hidden state from the previous time step s_{t-1} with matrix W. The output o_t gets computed by multiplying it with the matrix V. Afterwards the next time step s_{t+1} can be computed using the hidden state s_t. Non-linearity functions are again applied computing the hidden state as well as the output. The same parameters (the matrices U, W, V) are used at all time steps and do not change in time. The back-propagation algorithm can be applied in the same way as for a simple feed-forward network [LBH15]. Theoretical and empirical evaluations show that RNNs still lack the capability to learn store information for very long [BSF94].

The idea of a LSTM is to correct the lack of being able to store information for a long time by replacing the hidden neurons with ones that have a memory functionality [HS97]. A LSTM contains special memory blocks within the recurrent hidden layer. The memory blocks contain special memory cells with self-connections storing the temporal state of the network with special gates to control and influence the information flow.

Figure 2.6 shows a LSTM cell, that replaces the conventional cell of a RNN, displayed in Figure 2.5. The cell is defined by a set of vectors in $\mathbb{R}^d$: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden state h_t. The transition equations are defined as following:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \tag{2.4}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{2.5}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{2.6}$$

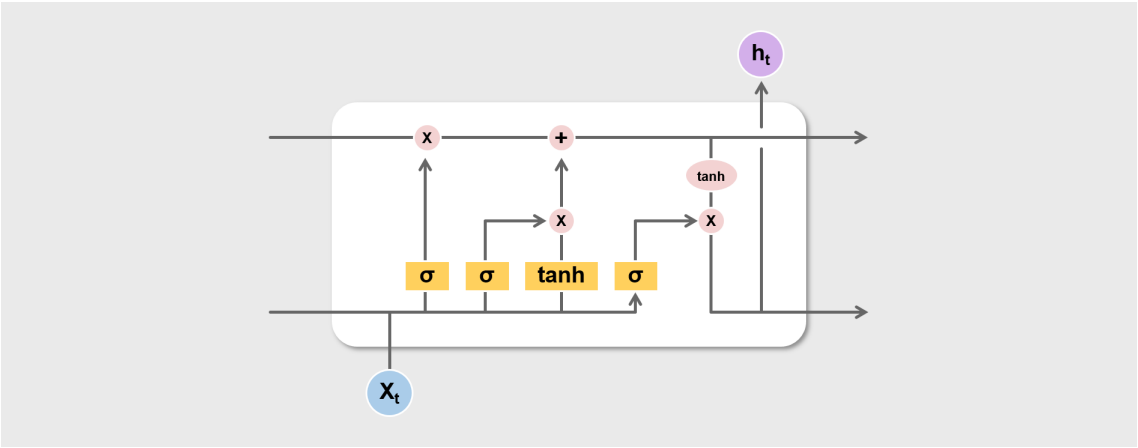

Figure 2.6: Long-Short-Term-Memory (LSTM).

$$u_t = \tanh(W^u x_t + U^u h_{t-1} + b^u) \tag{2.7}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{2.8}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.9}$$

Within these equations x_t defines the input at timestep t, σ defines the logistic sigmoid function (see Equation (2.3)) and $\odot$ stands for element-wise multiplication. Intuitively, the forget gate controls, how much the previous memory cell is forgotten, the input gate defines how much each unit is updated and the output gate controls how much the internal state is taking into account. The hidden state can be seen as the internal memory of the cell [TSM15]. In Figure 2.6 the information flow is shown, visualizing the equations.

Generative Adversarial Networks

GANs [GPM+14] are a powerful class of generative models, in which two networks are trained in a minimax two player game. A generator network produces synthetic data given some random noise vector and a discriminator network discriminates between the output of the generator and the true data. The generative model G tries to capture the data distribution, while the discriminative model D tries to estimate the probability that the received sample was produced by G. Figure 2.7 shows the two-player minimax game between the generator and discriminator. The generator needs to learn the distribution p_g over data x given input noise variables $p_z(z)$. The generator is defined by a differentiable function G, that maps the random noise vector to the data space as $G(z;\theta_g)$, where θ_g are the parameters of the generative network. The discriminator is defined as $D(x;\theta_d)$ and outputs a single scalar. $D(x)$ represents the probability that x came from the real data rather than p_g. The discriminator D is trained to maximize the probability to assign the correct label

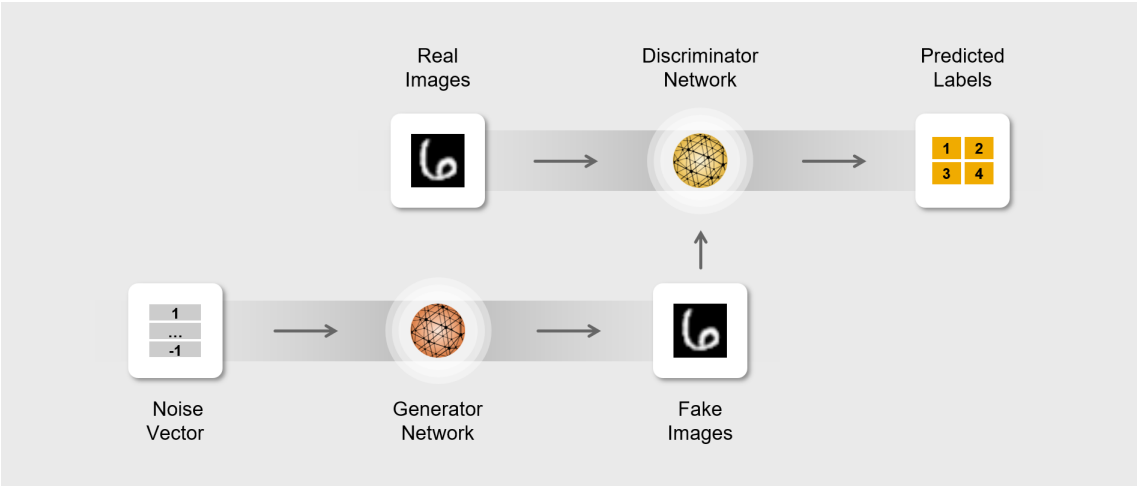

Figure 2.7: Generative Adversarial Network (GAN) training routine.

to the real training data as well as the samples produced by the generator G. The generator G is simultaneously trained to fool the discriminator D by minimizing $\log(1 - D(G(z)))$.

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.10}$$

The formalized problem, with value function $V(G, D)$ can be seen in Equation (2.10) [GPM+14].

If the discriminator is trained to optimality before each generator update, it often leads to vanishing gradients as the discriminator saturates. Goodfellow et al. [GPM+14] suggests that the generator should be instead trained to maximize $\mathbb{E}_{z \sim p_z(z)}[\log(D(G(z)))]$ to circumvent this difficulty. Even this adapted loss function can misbehave in a presence of a good discriminator [AB17].

Wasserstein GANs

GANs can produce very appealing data samples, but they are often hard to train. Improvements were made to stabilize the training of GANs [GAA+17]. Arjovsky and Bottou [AB17] provide an analysis of the convergence properties of the value function, that is being optimized by GANs. The solution they proposed is called WGAN [ACB17] and uses the Wasserstein distance to produce a value function with better theoretical properties than the original GAN. In a WGAN the discriminator is called critic and it must lie within the space of 1-Lipschitz functions, which is enforced by weight clipping.

The Wasserstein distance, also called *Earth-Mover* distance $W(q, p)$, is informally defined as the minimum cost of transporting mass in order to transform the distribution q into the distribution p. Thereby the cost is mass times transport distance. The WGAN value function is then defined as in Equation (2.11).

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{x \sim \mathbb{P}_r}[D(x)] + \mathbb{E}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] \tag{2.11}$$

$\mathcal{D}$ is the set of 1-Lipschitz functions and $\mathbb{P}_g$ is the model distribution defined by $\tilde{x} = G(z)$ and $z \sim p(z)$. In this case, given an optimal critic (it is not called discriminator, because it does not classify), minimizing the value function with respect to the generator parameters minimizes $W(\mathbb{P}_r, \mathbb{P}_g)$ [ACB17].

The WGAN value function results in a critic function with better gradients than in the original GAN, leading to an easier optimization of the generator. Empirical evaluation observed that the sample quality correlates with the WGAN value function, which was not the case for GANs [ACB17].

The proposed solution in Arjovsky et al. [ACB17] to enforce the Lipschitz constraint on the critic, is to clip the weights to make them lie within a space of $[-c, c]$. The set of functions satisfying this constraint is a subset of k-Lipschitz functions for a k that depends on c as well as the critic architecture.

Gulrajani et al. [GAA+17] shows the problems and difficulties with weight clipping and proposes a solution called Gradient-Penalty (GP). The updated version of WGAN is called WGAN-GP. Before going into the details of WGAN-GP, the problems of weight clipping are shortly described.

Weight clipping in WGAN can often lead to optimization difficulties. Although these problems can be mitigated by using batch normalization, WGANs still often fail to converge. A critic trained with weight clipping tends to ignore higher moments of the data distribution and instead models very simple approximations to the optimal functions. Another problem remains the vanishing or exploding gradients, if the clipping threshold c is not well trained [GAA+17].

As mentioned before, Gulrajani et al. [GAA+17] proposed a solution called GP to enforce the Lipschitz constraint. A differentiable function is 1-Lipschitz if and only if it has gradients with norm at most 1 everywhere. Therefore, the gradient norm of the critics output with respect to its input gets constrained directly.

$$\mathcal{L} = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1)^2] \tag{2.12}$$

The new objective (loss function) can be seen in Equation (2.12). The first part describes the original critic loss and, including the penalty coefficient λ, the GP is added. Another key finding is, that no batch normalization is used in the critic, in contrast to prior GAN implementations, where it is used in both the generator and discriminator and is needed to stabilize training. Batch normalization changes the form of the discriminators problem by changing it from a single input to a single output to an entire batch of inputs to an entire batch of outputs. The penalized training objective would no longer be valid with batch normalization, because the norm of the critic's gradient is penalized independently to each input and not an entire batch. Therefore, to avoid this, batch normalization is removed from the critic, but instead layer normalization is used as a drop-in replacement [GAA+17].

2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) is a multivariate technique that analyzes data. The observations within the data are described by various inter-correlated dependent variables. Its aim is to extract the important information out of all variables and represent it as a set of new orthogonal variables,

known as principal components. Mathematically, PCA depends upon the eigen-decomposition of positive semi- definite matrices as well as the Singular Value Decomposition (SVD) of rectangular matrices [AW10].

In more detail the goals of a PCA are:

- extract the most important information from the data,
- compress the size of the dataset by retaining only the principal components (important information),
- analyze the structure of the variables.

The principal components are obtained as linear combinations of the original variables. The first principal component has the largest variance, i.e. it contains or explains the most information of the data table. The second component is computed with the constraint of being orthogonal to the first component and to have the largest possible information. This pattern continues for the other components. The values of these new variables for the observations are called factor scores, and these factors scores can be interpreted geometrically as the projections of the observations onto the principal components [AW10].

For further information or details, please see Abdi and Williams [AW10] or Wold et al. [WEG87].

3 Related Work

This thesis focuses on continual learning methods that are applied in a complex field like VQA. We divide our work in two parts, the application, which is VQA and the machine learning algorithm belonging to the field of continual learning, which gets evaluated.

3.1 Visual Question Answering

VQA has become more popular in the last years [AAL+15; GKS+17; ZGS+16]. It belongs to the category of multi-modal problems in computer vision. Antol et al. [AAL+15] provided a dataset containing 0.25M images, 0.76M questions and 10M answers. They tested numerous baselines and methods with their dataset and compared it with human performance. Goyal et al. [GKS+17] balanced the previously mentioned dataset and has approximately twice the number of image-question pairs. They collected complementary images in a way that every question in the dataset is associated with a pair of similar images that result in different answers. The idea was to reduce bias in the dataset and focus more on the visual information. In contrast, previous work like Geman et al. [GGHY15] considers questions generated from templates from a fixed vocabulary of objects, attributes and relationships between objects. It thereby limits the validity of creating a system that can answer arbitrary questions about images. Malinowski and Fritz [MF14] used a restricted setting with a small dataset containing only questions the answers to which come from 16 predefined basic colors or from around 900 object categories.

We use the dataset from Goyal et al. [GKS+17] and a model from Antol et al. [AAL+15] as our baseline.

Johnson et al. [JHM+17] proposed an alternative, synthetic dataset for VQA, named CLEVR. They claim that is has minimal biases and annotations, which describe the images. It is designed to make a model learn the required reasoning abilities and test them to provide insight into strengths and limitations of state-of-the-art methods. The question representations allow to divide the dataset into different parts, for example question type, relationship type or question topology. Comparing the performance with the information of the different types allow for a better understanding of the reasoning capabilities of VQA systems. The advantage is that CLEVR controls biases that have been found in VQA datasets [GKS+17], which can be used by a neural network to answer questions without visual reasoning. Goyal et al. [GKS+17] tried to reduce the biases in their second version of the dataset.

Kuhnle and Copestake [KC17] proposed a VQA dataset that can be generated automatically during runtime to the experimenter's specifications. It is designed to evaluate multi-modal deep learning systems with a focus on formal semantic style generalization capabilities. They show that the Shapeworld dataset can be used to investigate what models learn with respect to multi-modal language understanding.

3.2 Continual Learning

Despite advances in DL, neural networks can only learn multiple tasks when these tasks are trained jointly. Otherwise neural networks suffer from so called catastrophic forgetting. The ability of learning multiple tasks sequentially is called continual learning and has gotten increasingly more attention in the past [AS19; HVD15; KGL17; KMA+18; KPR+17; LCG+19; LH17; MMPR19; PKP+19; RRD+16; SLKK17; WHL+18; YYLH17].

Continual learning can be divided into three different types of approaches: regularization-based, dynamic architectures and memory replays [AS19; PKP+19].

The first method, namely regularization-based approaches, constrains weight updates in order to avoid ”forgetting” of previously learned tasks and maintain the knowledge. Kirkpatrick et al. [KPR+17] proposed such a regularization-based approach, called Elastic-Weight-Consolidation (EWC). EWC is selectively slowing down learning on important weights of a previously learned tasks. It estimates the importance of each weight and adapts the updates accordingly. Another regularization method is distillation of already learned knowledge to a new model, introduced by Hinton et al. [HVD15]. It enables the model to learn information about the previous tasks and the current task at the same time. Learning Without Forgetting (LWF) [LH17] uses knowledge distillation to transfer the knowledge from a large, highly regularized model into a smaller model. A drawback is that the training time for one task linearly increases with the number of tasks learned so far [PKP+19]. Fernando et al. [FBB+17] developed PathNet, a neural network algorithm that uses agents embedded into the network to discover the important parts of the neural network that can be used to learn new tasks. To avoid catastrophic forgetting, they freeze task-relevant paths.

The second method is about dynamic architectures of neural networks. Rusu et al. [RRD+16] introduced progressive neural networks to avoid catastrophic forgetting and learn new information. A new neural network, referred as *column* in the paper, gets instantiated for each new task. Lateral connections to previously learned features leverage prior knowledge when learning a new task. Yoon et al. [YYLH17] proposed a Dynamically Expanding Network (DEN) that increases the number of trainable parameters to achieve the ability to continually learn new tasks. It is trained online and retrained selectively, which expands the network’s capacity using group sparse regularization to decide how many neurons to be added to each layer.

The third method is focused on memory replays, also known as rehearsal. The core idea is to keep samples from previous tasks and play them back during training of new tasks to retain knowledge. Instead of keeping samples from previous tasks, generative memory replays can avoid the requirement of storing old data and generate similar data samples when needed instead. This approach is referred as pseudo-rehearsal. Shin et al. [SLKK17] created an approach inspired by the CLS that uses generative replays as memory replays to retain previously learned knowledge. Training data from previously learned tasks is sampled in form of generated pseudo-data and combined with data from the new task. They defined various continual learning tasks using the MNIST and Street View House Number (SVHN) datasets. MNIST is a dataset containing handwritten digits [LCB10]. SVHN is a dataset containing images of house numbers extracted from [NWC+19]. It is not necessary to replay data samples from previous tasks for experience replay and therefore avoiding the memory constraint. For learning independent tasks, they trained the handwritten digits from MNIST and random permutation of it continually. For learning new domains, they trained MNIST and SVHN sequentially. They also trained to learn new classes, where they split all classes

into pairs of two and trained their neural network sequentially. Classes 0 and 1 get trained initially, 2 and 3 afterwards, and so on. These continual learning settings have been successful. Lesort et al. [LCG+19] found that generative replay methods outperform all other continual learning methods. Generative replay works well on MNIST and Fashion MNIST but training generative models on CIFAR-10 is particularly instable and remains a challenge. Wu et al. [WHL+18] also studied memory replay for continual learning. Their experimental results show that generative replay can work on MNIST, SVHN and LSUN. The generated images are of comparable quality and mitigate forgetting of previously learned categories.

We focus on memory replays, i.e. on generated replays using GANs, because the idea of a CLS copied from our nature appears to be a natural way of achieving lifelong learning.

3.3 Generative Adversarial Networks

Goodfellow et al. [GPM+14] initally proposed GANs for estimating generative models via an adversarial process. Two models are trained simultaneously, which corresponds to a minimax two-player game.

Recent advances improved GANs in terms of training stability using the Wasserstein distance [ACB17; GAA+17], as an alternative way of training GANs. Arjovsky et al. [ACB17] introduced WGANs and made progress towards stable training of GANs. Problems like mode collapse are mitigated and meaningful learning curves, which can be used for debugging or hyperparameter searches are introduced. Furthermore, they provided theoretical work on other distances between distributions, including the Total Variation (TV) distance, Kullback-Leibler (KL) divergence, Jensen-Shannon (JS) divergence and Earth-Mover (EM) distance (known as Wasserstein distance). They show that TV, JS and KL distances are not feasible for learning distributions. Concluding, they propose a practical approximation of optimizing the EM distance.

However, WGANs sometimes still produce poor sample quality or fail to converge. Gulrajani et al. [GAA+17] proposed an improved version of WGANs with gradient-penalty instead of weight-clipping. A WGAN-GP enables stable training with almost no hyperparameter tuning. They claim to achieve high quality generations on CIFAR-10 and LSUN bedrooms.

4 Approach

This section focuses on our approach derived from the previously described information. It provides a high-level overview about our core ideas before we go into detail with the experiments.

We want to build a VQA system that is able to acquire knowledge during its lifespan. In Chapter 2 we showed the background of work. We went into detail about the problem of continual learning as well as visual question answering. We also described the background information about the state of the art techniques to address the problem using machine learning, more specifically deep learning methods. In Chapter 3 we showed state of the art approaches and experiments of related works. Many experiments and progress have been made within this domain using different kinds of datasets.

As a consequence of our investigation, we decided to use generative replay as continual learning method, in particular we use a WGAN-GP as GAN through our work. WGAN-GP tend to be more stable during training and produce high quality data samples. We want to propose the research question, if it is possible to enable continual learning with generative replay, using a WGAN-GP, in the domain of VQA. We investigate multiple datasets, namely the VQAv2 dataset [GKS+17], the CLEVR dataset [JHM+17] and the Shapeworld dataset [KC17].

As a first step, we show a proof of concept using a toy dataset like MNIST. We want to make sure that our approach is working, before addressing more complex datasets in the domain of VQA. Within this part of our work, we step wise increase the difficulty: starting with implementing a task-solver for MNIST, followed by verifying if a WGAN-GP is capable of generating samples comparable to the original dataset, until training our MNIST task-solver continually with each handwritten digit individually.

The next step is to apply the same method using the VQAv2 dataset and thereby making the transition into the domain of VQA. Again, we start by creating a VQA-solver to test the performance in general. Afterwards, we define continual learning tasks for the VQAv2 dataset. We improve the task-solver by enabling training of different tasks continually.

The CLEVR dataset is the second dataset that we use out of the domain of VQA. CLEVR is a synthetic dataset that will be explained in more detail inside the chapter about the corresponding experimental results (see Section 5.4). We want to apply the same procedure as previously described for the VQAv2 dataset.

The last dataset that we investigate out of the VQA domain, will be the Shapeworld dataset. It is an synthetic that can be generated during runtime and is highly flexible in its complexity. More details follow in Section 5.5, where we describe our results.

4.1 Generative Replay

In this section, we explain our generative replay setup with the corresponding terminologies. We train a sequence of tasks sequentially; $T = (T_0, T_1, ..., T_N)$ with a total of N tasks. The individual, continual learning task definition for each dataset is given in the corresponding section in Chapter 5.

In our setup, we define a task solving model (often referred as task solver in this thesis), which is optimized to achieve a high accuracy on the dataset. Initially, it gets evaluated with the complete dataset to prove its effectiveness. The next step is to split the dataset according to the continual learning task definition into tasks T_0 to T_N.

Our model gets optimized to solve task T_i by being trained on data D_i from which training examples (x_i, y_i) are drawn.

To retain the knowledge of the previously learned data distribution D_i in the task solver, we feed in real-like samples from a generative model (as mentioned before, we use a Wasserstein GAN with gradient-penalty). Generative replay is not required for the first task T_0, since there is nothing to "remember" yet.

However, the generative model needs to reproduce all data distributions D_i - it gets optimized on D_i after the task solver, so it can be used to generate pseudo-data when the following task is being trained. To retain the knowledge of the generative model of previously seen data distributions D_0 till D_{i-1}, we samples pseudo data and merge it with the data D_i to optimize the generator on all data distributions D_0 to D_i. The generative model, which is optimized to generate samples till task T_i, is then used to retain the knowledge of previous tasks for the task solver when the task solver is trained with task T_{i+1} on data D_{i+1}. During training of the task solver on D_i, we sample from the generative model and feed it into the network.

5 Experimental Results

The goal of this thesis is to enable CL for VQA. The approach itself is divided into four parts. First of all, the idea of using generative replay to overcome catastrophic forgetting in neural networks is implemented using a toy dataset, i.e. using the MNIST dataset for handwritten digit recognition. In the next step, the resulting techniques are applied for the VQAv2 dataset [GKS+17]. To reduce the complexity, we apply our approach on CLEVR, a synthetic dataset. Each image contains different objects with specific properties, which the questions are aimed at. At last, we investigate a dataset called Shapeworld, also a synthetic dataset that generates data samples during runtime. We consider Shapeworld to be even less complex than CLEVR. All datasets belong to the multi-modal field of VQA and aim to test a system regarding generalization and language understanding abilities. Afterwards, we compare and analyze the results.

5.1 WGAN-GP

This section describes the implementation of our WGAN-GP. The architecture and hyperparameters are derived from Arjovsky et al. [ACB17] and Gulrajani et al. [GAA+17]. We use deep convolution networks as generator and critic - many other experiments failed to achieve similar results, for example MLP's as critic, generator or both.

Figure 5.1 shows the generator that is used for different domains through this work. The amount of

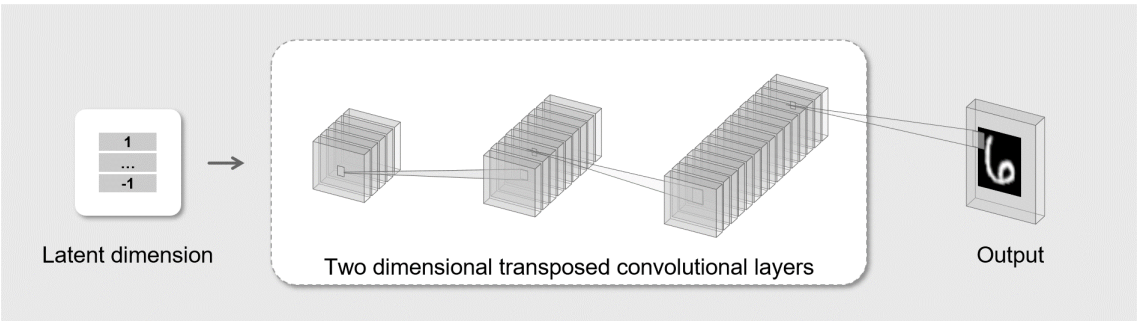

Figure 5.1: Architecture of the generator of our WGAN-GP.

layers differ from task to task depending on the desired size of the image. The input of the network is an arbitrary vector of size 128, followed by different transposed convolution layers. Each layer uses batch normalization and ReLU as activation function. The number of channels is cut in half between transposed convolution layers. The kernel size of the filters is set to four and a stride of two is used with zero padding of one. The output of the generator is an image.

Figure 5.2 shows the critic that is used for different domains through this work. An image, real or

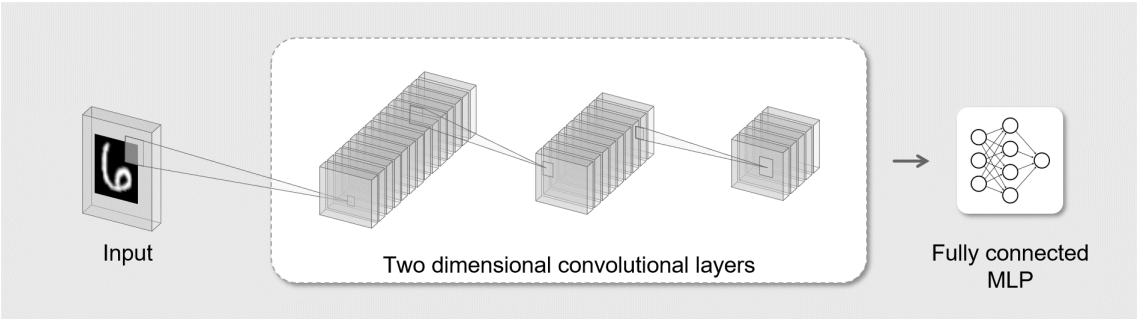

Figure 5.2: Architecture of the critic (discriminator) of our WGAN-GP.

generated, is fed into the network as input. Our critic contains multiple convolutional layers - the amount differs from task to task, in the same way as for the generator. Despite that, leaky ReLU is used as activation function. The critic uses layer normalization instead of batch normalization. The number of channels is doubled between convolution layers. The kernel size of the filters is set to four and a stride of two is used with zero padding of one. The output of the critic is a scalar value that determines the difference between real and generated data. The architecture looks familiar compared to the generator, but in a reversed way. Instead of creating an image, it extracts information to map the generated image distribution to the real distribution.

5.2 MNIST

This section describes training a neural network that classifies handwritten digits of the MNIST dataset while trained in a continual manner. We define a continual learning task for MNIST as follows: each digit represents an individual task. Tasks are trained in a sequential order. The objective is to achieve a comparable performance over all tasks similar to training the model with the full dataset containing samples of all tasks.

5.2.1 MNIST-Solver

The first step towards training a neural network with the MNIST dataset continually, is to create a network that achieves a good performance on the dataset. Figure 5.3 shows the architecture of

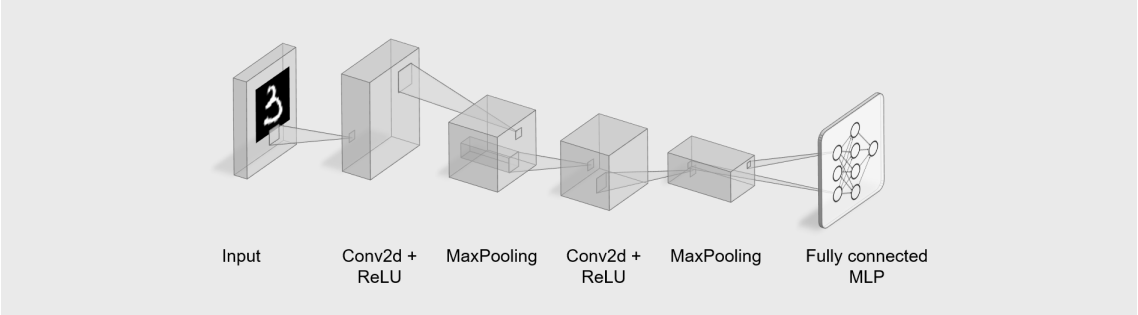

Figure 5.3: Architecture of the MNIST Solver.

our MNIST Solver, which is a CNN. The input is an image of size 28x28, which gets fed into the network. It contains of two 2-dimensional convolution layers, each followed by a max pooling layer. The filters of the convolution layers have a kernel size of 5 and stride of 1. The max pooling layers have a kernel size of 2 and stride of 2. Afterwards, the image features get flattened and fed into a fully-connected layer to classify the image into the digits zero to nine. We used a batch size of 64, a learning rate of 0.001 and trained for 5 epochs. The images were normalized before training.

Figure 5.4 shows the performance of the previously defined model. The first epoch (epoch zero)

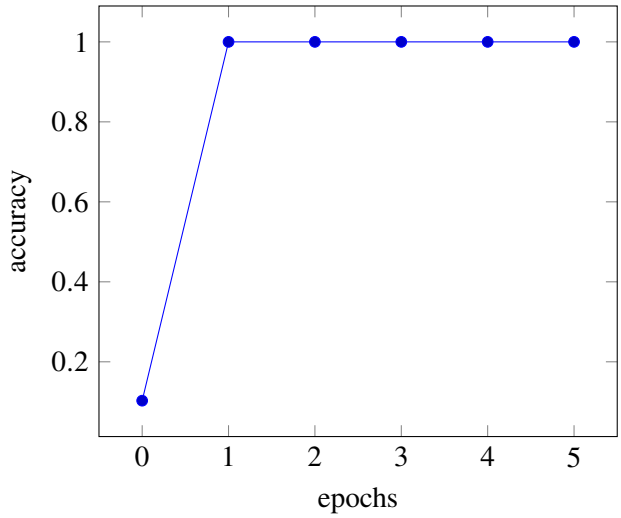

Figure 5.4: Accuracy of the MNIST-Solver on the test dataset.

tested the model's performance on the test dataset before any training had happened. The accuracy is at 10%, which corresponds to classifying the input arbitrarily. After training one epoch, the accuracy of the model jumps to 100% and stays at the same level for the following training epochs. Our defined task-solver for the MNIST dataset is able to classify all handwritten digits with an accuracy of 100%.

5.2.2 Generating Images of MNIST

After successfully creating a model that scores a good performance on the MNIST dataset, we need to be able to generate samples similar to the dataset to train the model sequentially with each digit. The idea is that the GAN produces generated images of handwritten digits that already have been trained to retain the knowledge of classifying them. The first step is to determine, if our defined WGAN-GP can generate images similar to the original dataset. Figure 5.5 displays the results of our defined WGAN-GP after training it for 400 epochs. We state that the produced samples are at least comparable to the original images.

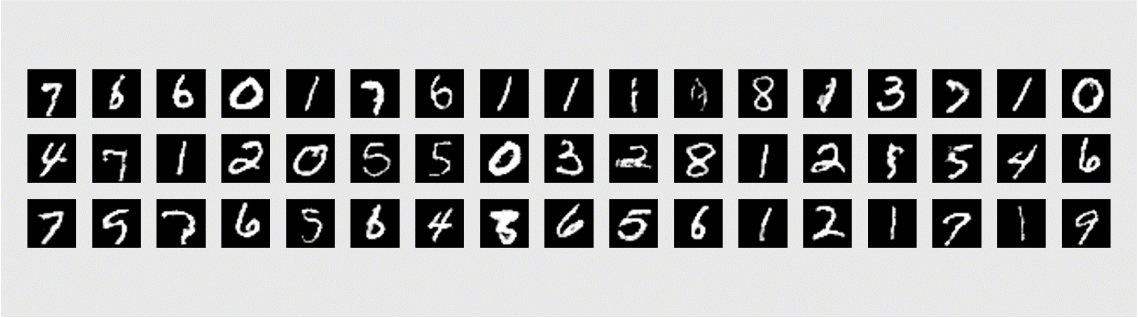

Figure 5.5: Samples generated by the WGAN-GP trained with all digits of the MNIST dataset.

5.2.3 Catastrophic Forgetting

In this section we show that our proposed solver for MNIST suffers from catastrophic forgetting. Figure 5.6 displays the performance of the model when we do not use a continual learning method. The performance is measured in two different ways: the first one calculates the accuracy on all test-datasets from already seen digits and the second one calculates the accuracy on all test-datasets from all digits, i.e. testing on digits that have not been seen yet. The accuracy, calculated on all

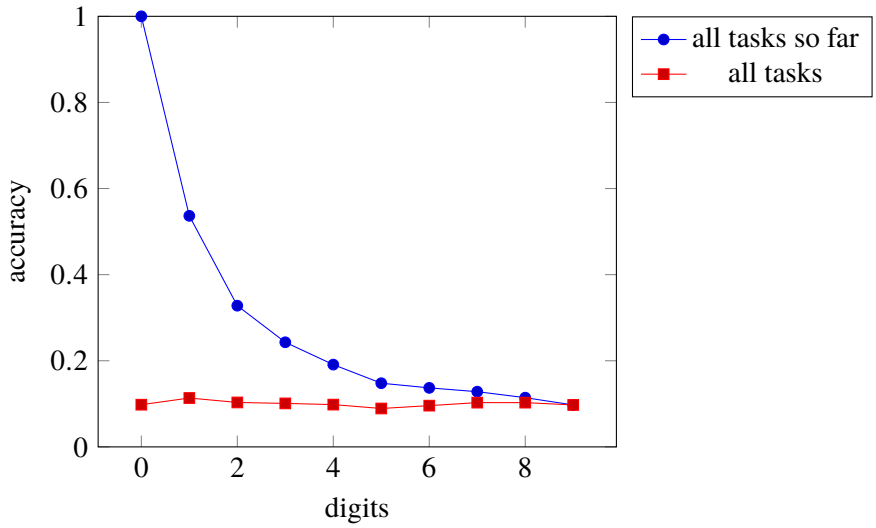

Figure 5.6: MNIST with continual learning - accuracy of all tasks seen so far.

tasks, stays around 10%, since our model can only classify the current task/digit correctly. The validation on tasks seen so far decreases with each newly learned task, because the information of previously seen digits is forgotten.

Figure 5.7 gives more details regarding each training task. We observe that each training task can be perfectly classified when our model is trained with the corresponding dataset. With the next time step, when the next task (digit) is trained, the knowledge of classifying the digit is completely wiped out, resulting in an accuracy of 0%.

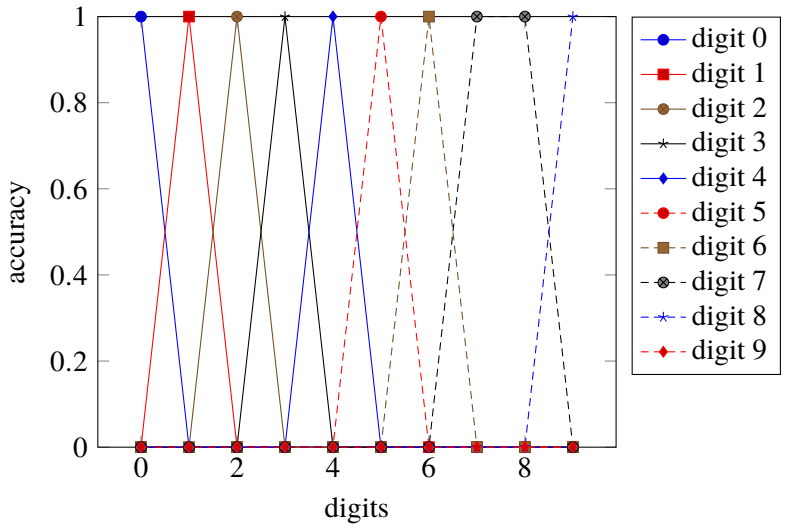

Figure 5.7: MNIST without continual learning - accuracy of all tasks or digits.

Algorithm 5.1 MNIST training algorithm

```
for all d ∈ 𝒟 do
    if d ≠ firstTask then
        samplesGenerated ← GAN.SAMPLE
        labels ← MNISTSOLVER(samplesGenerated)
    end if
    MNISTSolver ← MNISTSOLVER.TRAIN(d, samplesGenerated, labels)
    GAN ← GAN.TRAIN(d, samplesGenerated)
    for all d ∈ 𝒟 do
        MNISTSOLVER.TEST(d)
    end for
end for
```

5.2.4 Continual Learning for MNIST

We showed that our WGAN-GP is capable of generating images with comparable quality as well as that catastrophic forgetting exists in our neural network. The following step is to use samples as generative replays to train all handwritten digits sequentially.

Algorithm 5.1 shows the training routine of our approach to train each handwritten digit continually. $\mathcal{D}$ defines the set of all tasks, in this case all handwritten digits from zero to nine. We differentiate between two states of training: the first task and all others. At the first training task, we do not need generative replays for MNIST-Solver, because our model does not need to remember a task so far. We start by training the task-solver initially and move on to train the WGAN-GP to reproduce samples from digit zero. After having trained both, we evaluate the accuracy on all validation datasets for each digit. For all following tasks, we generate samples from our WGAN-GP and create labels for them by using our task-solver before training a new task/digit. At that point the task-solver is able to classify all samples generated by the WGAN-GP and we can use the prediction as label

for the following training steps. These generated samples get mixed up with the original data from the current task. Afterwards we use the generated samples also to train our WGAN-GP to produce samples from the current task as well as from previous tasks. We repeat this routine for all digits d in $\mathcal{D}$.

Figure 5.8 shows the performance of the proposed model while training the handwritten digits sequentially. The accuracy is displayed in two different ways: the first one being on all tasks (digits)

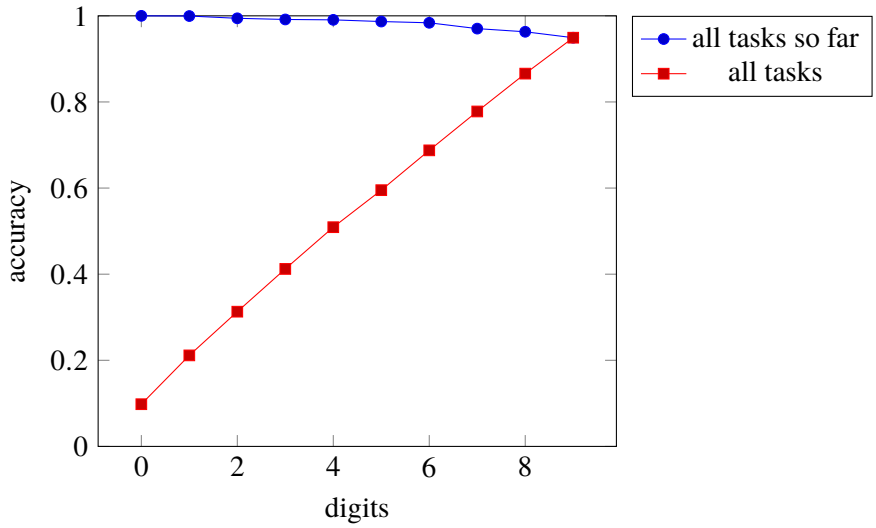

Figure 5.8: MNIST with continual learning - accuracy of all tasks seen so far.

seen so far and the second one being on all tasks. The accuracy calculated on all tasks uses the full test-dataset to measure the performance. The accuracy increases as expected by around 10% at each new digit trained, reaching 94.9% at the end. The accuracy calculated on all tasks so far uses all test-datasets of digits that the model has been trained on yet. The model can keep its performance on previously seen digits and does not drop to zero as we have seen when no replays are used (see Section 5.2.3).

Figure 5.9 shows the performance on each digit individually. The accuracy of each digit is at 0%, when the model has not seen any samples of it. After training a digit, the accuracy jumps close to 100% and can be kept while the continual training of digits continues. After training all tasks/digits the neural network is able to classify all handwritten digits.

5.2.5 Conclusion

We started this section by implementing a solver for the MNIST dataset. The following step was to show that our WGAN-GP is able to generate fake images, which look close to the real ones. We showed that catastrophic forgetting occurs within our model when we train each handwritten digit sequentially. Afterwards we implemented our generative replay approach to verify, if our continual learning methods overcomes ”forgetting” previous knowledge.

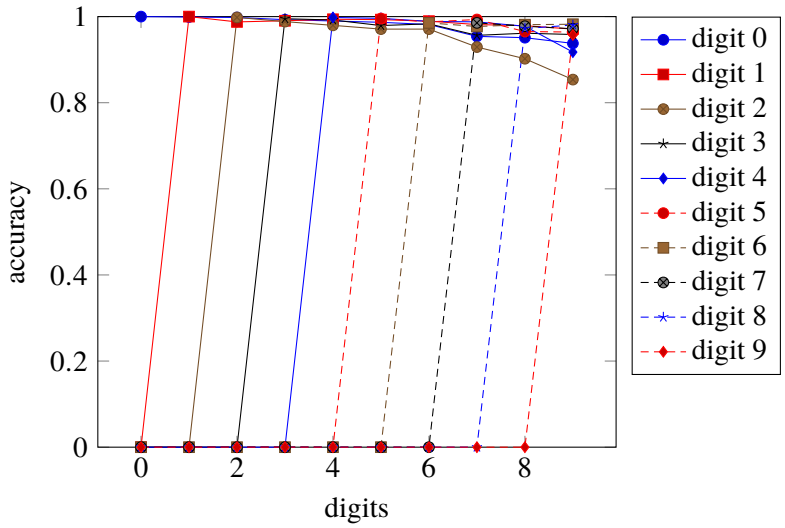

Figure 5.9: MNIST with continual learning - accuracy of all tasks or digits.

Section 5.2.4 shows that continual learning with generative replay works for the MNIST dataset. The accuracy over all tasks can be kept throughout the training with each digit individually. Catastrophic forgetting, as it can be seen in Section 5.2.3, does not occur anymore. In detail, when we compare Figure 5.8 to Figure 5.6, we observe that the accuracy of all tasks increases incrementally when trained with generative replays instead of staying at 10%. The validation on all tasks seen so far states the same conclusion: the accuracy stays close to 100%. If we compare Figure 5.9 to Figure 5.7, the graphics support our conclusion as well - the accuracy of no task drops towards 0% as it happens, if the baseline model does not use generative replays. Table 5.1 summarizes the

Accuracy	Generative Replay	No Generative Replay
After first task	100%	100%
After all tasks	94.9%	9.7%

Table 5.1: Comparison of generative replay vs. no generative replay on the MNIST dataset.

results of training each handwritten digit of MNIST sequentially.

As a next step, we move away from the uni-modal problem of MNIST to the multi-modal domain VQA.

5.3 VQA

This section describes the approach to enable continual learning in a VQA environment. We define a continual learning task with respect to the VQAv2 dataset [GKS+17] as follows: the dataset gets split into different question types. We define a question type by the word the question begins with and we use the nine most common question types within the dataset.

Figure 5.10 shows the data distribution of the VQAv2 dataset when we split the questions into groups by their starting word. The training dataset contains a total of 443,757 questions. Around

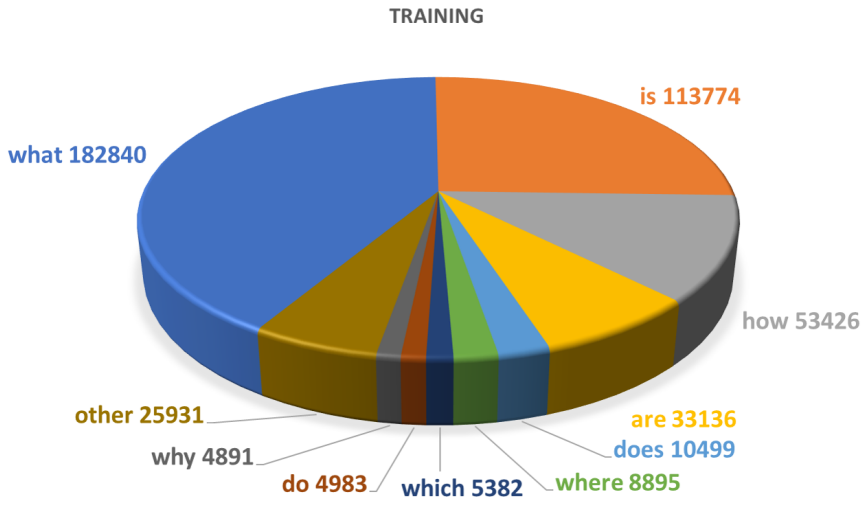

Figure 5.10: VQAv2 [GKS+17] split training data distribution.

40% start with "what", an open-ended question, followed by around 25% 'is'-questions, which are answered with "yes" and "no". All questions beginning with any other word are grouped together under "other".

Figure 5.11 shows the data distribution of the corresponding validation dataset, with a total of 214,354 questions. The distribution differs slightly, there are as many questions starting with "is" as

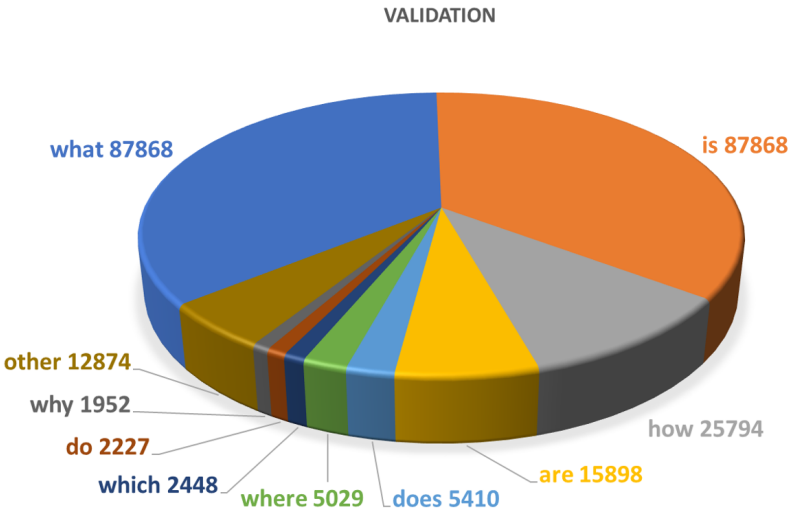

Figure 5.11: VQAv2 [GKS+17] split validation data distribution.

with "what". The percentages of the other question types almost stay the same.

Our focus is on the nine largest question types containing a mix of open-ended and ”yes”/”no” questions. We omit all ”other” question types. We train our models in a descending order - from the largest dataset of questions types to the smallest. The structural difference of these question types helps measuring the success or failure of our continual learning method, because our hypothesis is that open-ended question and ”yes” and ”no” questions influence each other positively. The order of our training routine for the different questions types looks like following: what, is, how, are, where, does, which, why and do. In other words, we trained question type ”what’ at task 0, ”is” at task 1, ”how” at task 2 and so on for each figure within this chapter.

5.3.1 VQA Baseline

Figure 5.13 shows our VQA baseline model, which is reconstructed from Antol et al. [AAL+15]. It contains of three parts: a CNN to extract the image features, a LSTM to get the question features and a MLP that combines both feature vectors to finally classify the 1000 most common answers. We use the VGG19-CNN of Simonyan and Zisserman [SZ14]. It contains of 19 deep convolutional

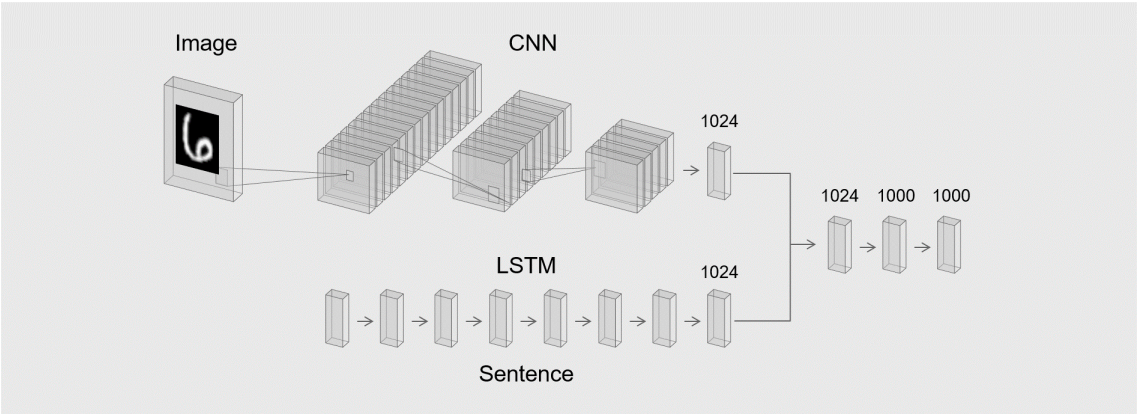

Figure 5.12: Architecture of the VQA baseline model. Adapted from [AAL+15].

weight layers. The output layer is modified to return a vector of size 1024 instead of 1000. Table 5.2

Hyperparameter	Value
Learning Rate	0.0008
Optimizer	Adam
Word Embedding	300
Output	1000
LSTM Layers	2
LSTM Hidden Dimension	512
Dropout	0.5
Activation Function	tanh

Table 5.2: Hyperparameters of our VQA Baseline

contains all hyperparameters that have been used to train the baseline. The activation function refers

to the activation function used in all hidden layers - the output layer used a softmax layer, as it was described in the original paper. Since no optimizer was mentioned, we used Adam. Furthermore, we used early stopping to find the optimal number of training epochs. We decided to use this model as baseline, because we can preprocess the images and generate the combined feature space of the images and questions.

We normalized the images before training, and we used a pre-trained VGG-model to calculate all image-features before training to speed up the training process.

Figure 5.13 displays the accuracy on the validation dataset during training. The performance peaks

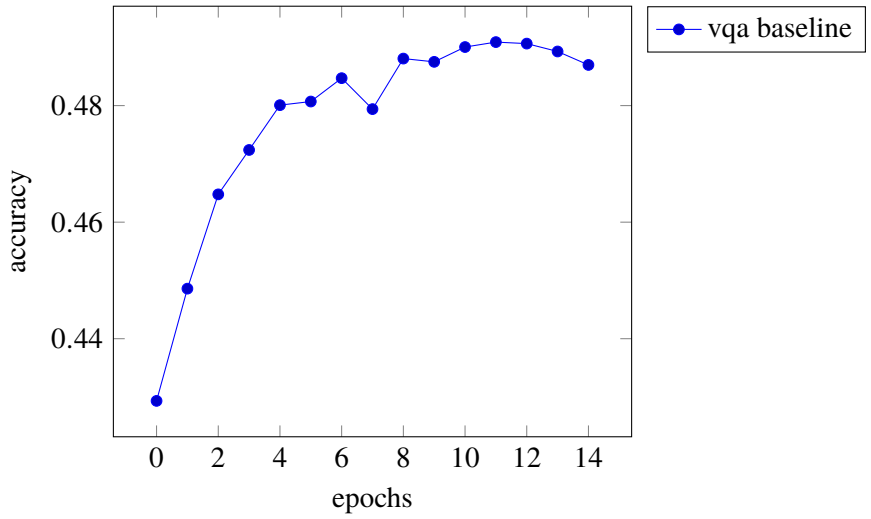

Figure 5.13: Accuracy of the VQA baseline model on the test dataset.

at 49%. When we train our baseline model with generative replays to enable continual learning, we measure the success with the achieved accuracy.

5.3.2 Catastrophic Forgetting in VQA

This section shows that our VQA baseline model suffers from catastrophic forgetting. We trained the model in the sequential order that has been described at the beginning of this section and calculated the accuracies for each question type as well as for all question types consolidated, i.e. all question types that already have been trained so far to compute the overall performance.

Figure 5.14 shows the result of training the baseline model sequentially with different question types and not using any continual learning method. After training a question type, all other question types get tested as well. The first two question types that have been trained are questions that begin with 'what' and 'is'. The main difference between these two question types is, that 'what'-questions are open-ended in contrast to 'is'-questions, which are answered with 'yes' and 'no'. After training the first question type (questions beginning with 'what'), our model learns to answer open-ended questions - validating the accuracies of other open-ended questions show that the baseline model can also answer questions correctly from other open-ended question types. The following iteration trains questions that begin with 'is'. Now, the baseline needs to learn to answer with 'yes' and 'no'.

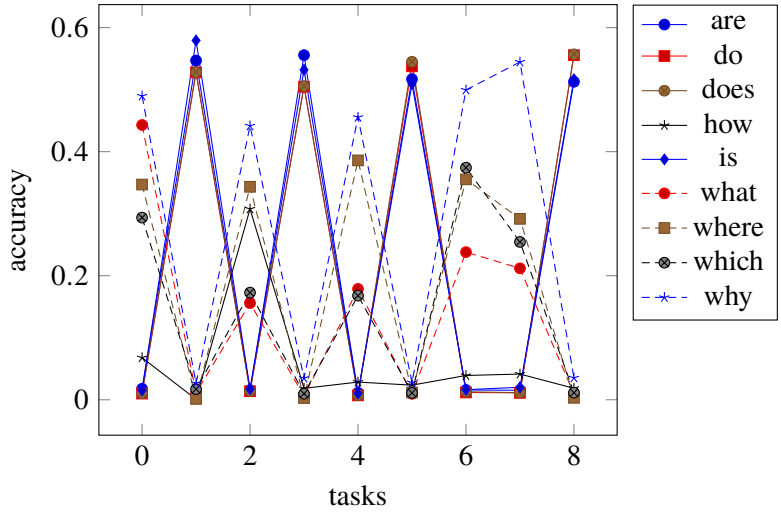

Figure 5.14: Catastrophic forgetting of our VQA baseline model.

The data show that the accuracies of all other 'yes'/'no' question types also increase. The accuracies of open-ended questions immediately decrease to 0%. The model does not retain already learned knowledge. This pattern continues through the training process, the performance of previously learned question types decrease to 0%, if an open-ended question type follows upon a 'yes'/'no' question and vice versa.

Figure 5.15 shows the accuracy of the consolidated validation datasets of question types that the model has been seen so far. Similar to the MNIST domain, the VQA baseline is unable to keep its

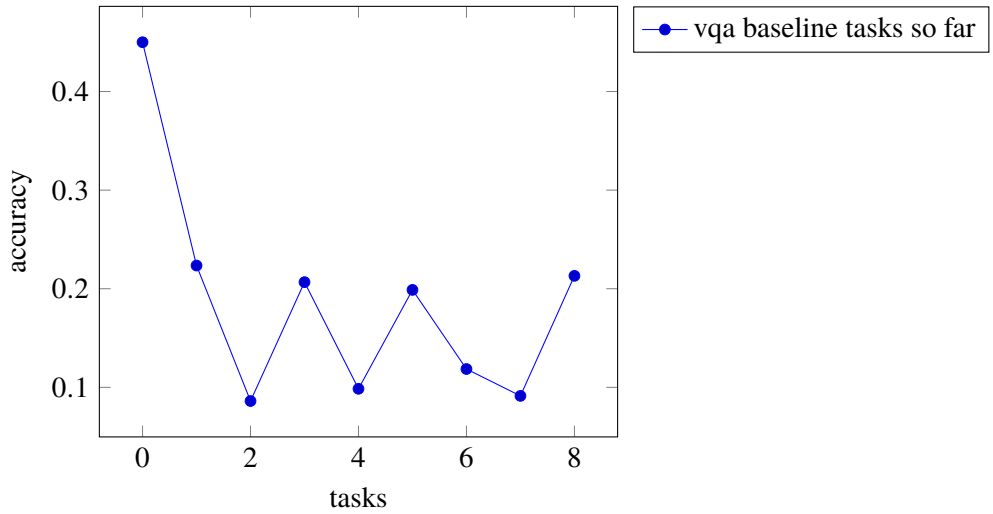

Figure 5.15: Catastrophic forgetting of our VQA baseline model. Accuracy of validation datasets together.

accuracy across tasks. The performance after learning a new question type drops in most cases. Small increases of the model's performance after some tasks occur, because the question types

influence each other. After training a 'yes'/'no'-question type, the model learns to answer with 'yes' and 'no' only and increases the accuracy of other question types that are answered with 'yes' and 'no' as well. We conclude that the baseline model suffers from catastrophic forgetting.

5.3.3 Continual Learning in VQA

This section investigates, if generative replay as continual learning method helps to overcome catastrophic forgetting. VQA is a more complex domain than our previous domain MNIST. To reduce complexity and increase chances of success, we decided to generate the combined feature space of the image and question pairs instead of generating images and questions separately. To generate an image of the VQA dataset itself would be a huge challenge, but also generating a meaningful question with respect to the generated image seems too hard for the next step.

We use the same architecture of our generative model that we have used for the MNIST domain with a small change. The objective is to generate the combined feature vector of image and question, which is of size 1024. We increase the size of the generated image by our WGAN-GP to 32 times 32 and flatten it to get a vector of size 1024 as it is needed. During the training process, we feed in memory replays into the MLP of our baseline to retain previously learned question types.

Algorithm 5.2 shows the training routine of our approach to train each question type continually. Q defines the set of question types that we have specified at the beginning of this chapter. Again,

Algorithm 5.2 VQA training algorithm

```
for all q ∈ Q do
    if q ≠ firstTask then
        generatedFeatures ← GAN.SAMPLE
        labels ← VQASOLVER(generatedFeatures)
    end if
    VQASolver ← VQASOLVER.TRAIN(q, generatedFeatures, labels)
    currentFeatures ← VQASOLVER.CALCULATEFEATURES(q)
    GAN ← GAN.TRAIN(currentFeatures, generatedFeatures)
    for all q ∈ Q do
        VQASOLVER.TEST(q)
    end for
end for
```

we differentiate between two states of training: the first question type and all others, which are trained continually in a sequential order. For the first iteration, we train the VQA-solver (our defined baseline model) with the training data belonging to question type q - at this step the generated features and corresponding labels are zero, because there is no task to remember. After we have optimized our model for recognizing the first task, we can use the model to generate the combined feature vector of questions and images. Now we can train our generative model to reproduce these feature vectors. The last step is to evaluate the accuracies of all question types individually. If we train a question type continually, we use the WGAN-GP to generate feature vectors of the previous tasks to feed them into the VQA-solver paired with training data from the current task/question

type. To update the generative model, we need to calculate the feature vectors of the current task again as well as the generated features from previous tasks to enable the WGAN-GP to produces samples from all tasks trained so far.

Figure 5.16 shows the results of training the baseline model with generative replay for each question type. Each open-ended question type drops to 0% after training a 'yes'/'no' question and vice

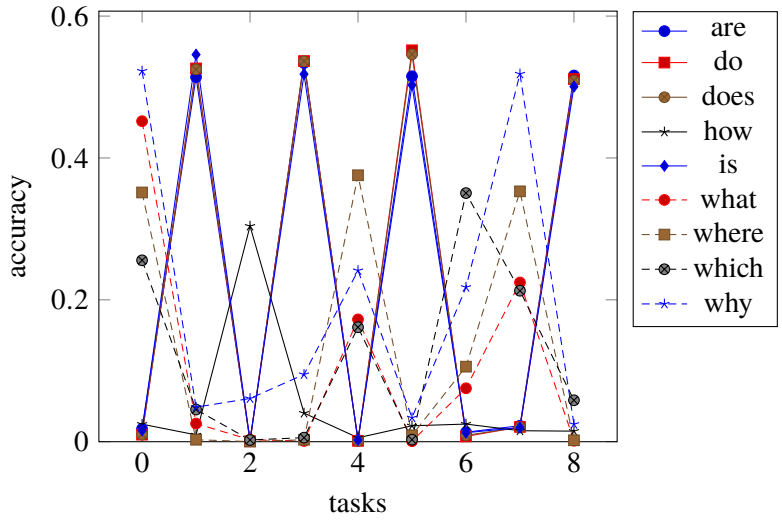

Figure 5.16: Accuracy of the VQA baseline model with generative replay.

versa. The result looks similar to Figure 5.14 in which no continual learning method was used. Our approach of generating the combined feature vector does not work for the VQA domain.

Figure 5.17 shows the performance of the baseline tested on all tasks trained till task t. The figure also contains of the graph of Figure 5.15 to make it easier to compare the baseline model with and without generative replay. The model's performance without is referred as catastrophic forgetting. The performance of the model behaves in the same way as if no generative replay would have been used.

Figure 5.18 shows a two-dimensional dimensionality reduction of the feature vectors using a principle component analysis. We plotted these two data distributions, because it is not intuitively possible to visualize the results of the generator otherwise. If we generate images, the results can be observed and evaluated by humans, which is not the case for feature generation. The dimensionality reduction supports our results of training the baseline with generative replay. The generated data distribution does not match the real data distribution. The generator could not capture and learn the real data distribution.

Figure 5.16 and Figure 5.17 show that generative replay did not help to overcome catastrophic forgetting in the domain of VQA with our task definition. The model is not able to keep already learned knowledge, i.e. answering a previously seen question type.

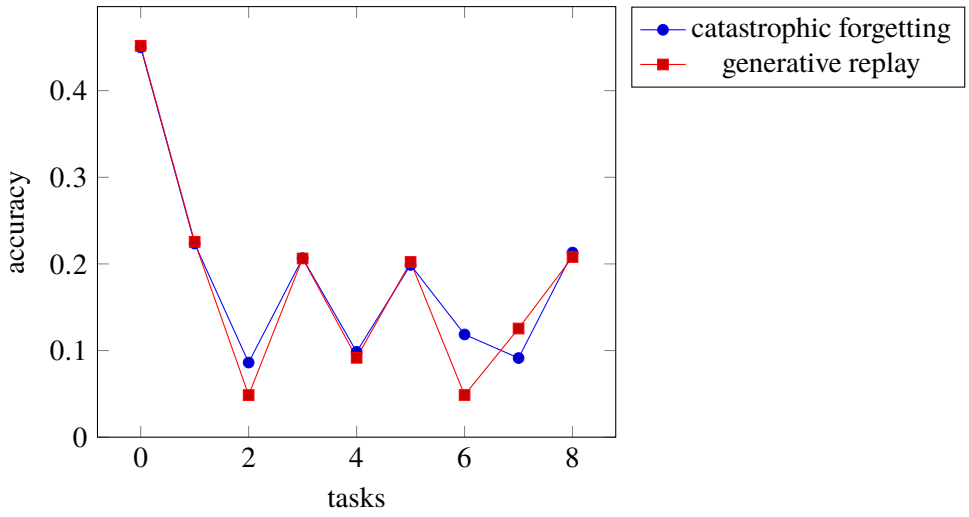

Figure 5.17: Accuracy of the VQA baseline model with generative replay tested on all test datasets of tasks seen so far.

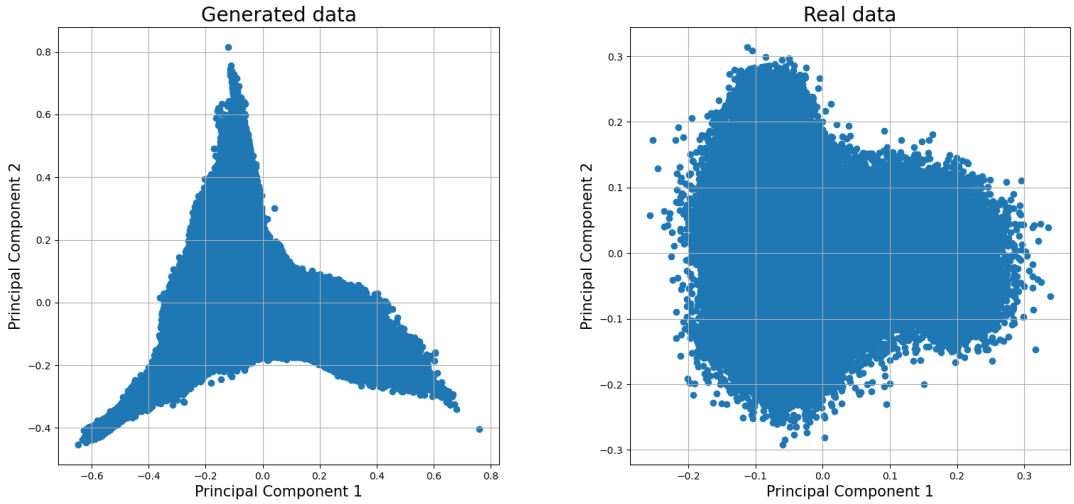

Figure 5.18: Caption

5.3.4 Split Question Types 8:1

Therefore, we change the continual learning task to train eight question types in advance and train only one in a continual way. The idea is, that this could lead to successfully train a question type continually.

Figure 5.19 shows the results of training eight questions in advance and one continually. The figure contains four graphs, two represent the performance of our model with generative replay and two without it (catastrophic forgetting). The accuracy at task $t = 0$ displays the result of training all questions types without questions beginning with ”do”. The accuracy concurs with the accuracy

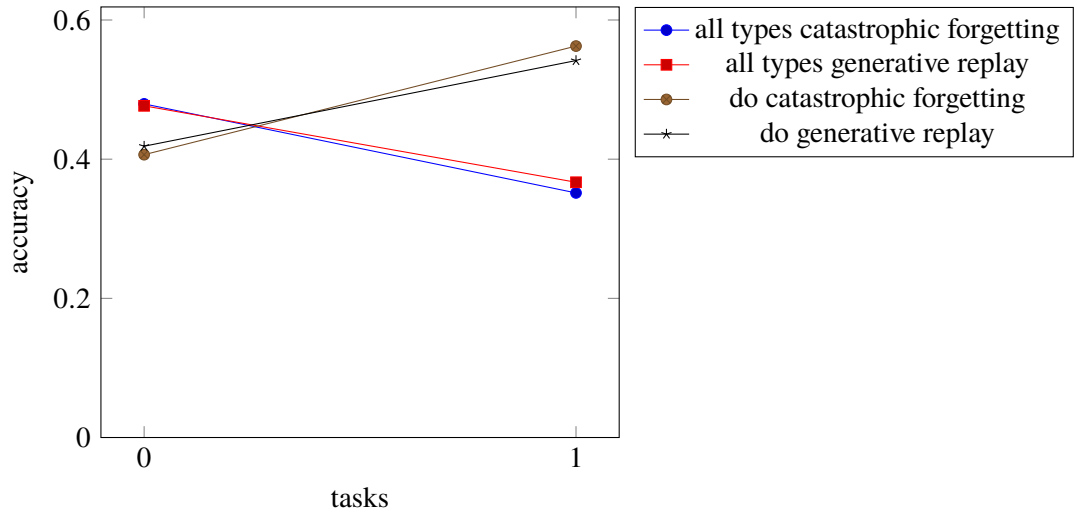

Figure 5.19: Accuracy of the VQA baseline model with generative replay training eight questions in advance and one continually.

of our baseline, if all question types are trained together. The accuracy of the ”do”-questions is around 40%, because the network already has learned to answer with ’yes’ and ’no’. The following training step uses the dataset of the ’do’-questions only and therefore its accuracy on the validation dataset increases to 56% in both cases. The interesting part is that there is almost no difference in training the ”do”-questions with or without generative replay. The performance drops to 36%. Our generative replays of the previous eight question types should have prevented the loss of accuracy, which is not the case. Even pre-training most of the dataset and only training one question type - as a matter of fact it is also the smallest dataset of all nine - in a continual learning manner did not help to enable continual learning in our VQA setting. Table 5.3 summarizes the consolidated

Accuracy	**Generative Replay**	**No Generative Replay**
After 8 tasks consolidated	47.6%	47.9%
After 9th task continually	36.6%	35.1%

Table 5.3: Comparison of generative replay vs. no generative replay on the VQAv2 dataset with a 8:1 split.

results and shows the performance loss, although using generative replays.

We narrow the problems of our results down to two major reasons: the accuracy of our VQA baseline is too low or the generator cannot capture and reproduce the feature vectors. We use the previous task-solver to generate the labels for the generated data samples, which are used to retain the knowledge from previously learned tasks. An accuracy of 50% produces only 50% correct labels, which is the case for our baseline. This error accumulates through tasks, the network adapts to wrongly classified data and cannot retain the performance. Compared to our MNIST task-solver, which has an accuracy of 100% and therefore never produces wrongly classified labels, this could lead to fail to train tasks continually.

In the following sections, we adapt our approaches to increase in terms of updating the VQA baseline and datasets to verify our hypotheses.

5.3.5 Reduced Output

This section focuses on reducing the output layer of the VQA model to increase its performance. The objective is to verify if the error accumulates and makes it impossible to train tasks continually.

Figure 5.20 shows the accuracy of the baseline model with a reduced output layer to 50 classes. The performance increases by around 17 percent points compared to the VQA baseline model.

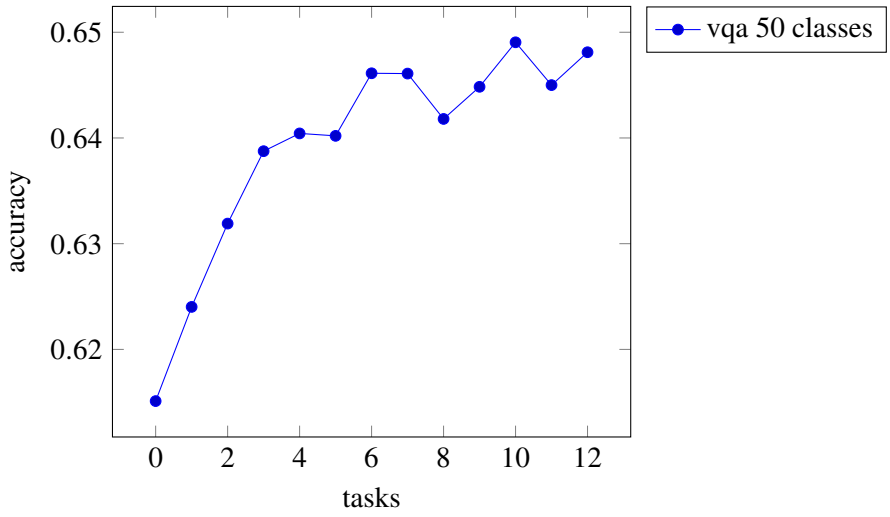

Figure 5.20: Accuracy of the VQA baseline with a reduced output to 50 classes.

The following step is to train all question types sequentially with and without generative replay to check, if the generated samples enable continual learning when the accuracy of our baseline is increased due to less classes that are predicted.

Figure 5.21 shows the performance of our VQA baseline model when we reduce the output layer to 50 classes and do not use generative replay. The initial accuracy of each question type is increased when the model only predicts the 50 most common answers instead of 1000. But it still suffers from catastrophic forgetting, training the next question types result in performance loss. For example, our baseline achieves an accuracy of 76% for the 'what'-questions but drops close to 0% when the model's trains on the following 'is'-questions. We showed that our baseline model with an output layer of size 50 still suffers from catastrophic forgetting. We can now move on and train the same model with generative replay and check, if a higher accuracy of the solver model helps to retain previously acquired knowledge.

Figure 5.22 shows the accuracy of all tasks, if our model is trained with generative replay. The accuracies of tasks decrease after training the other question types sequentially. Our generative replays could not avoid catastrophic forgetting. The learning curves look similar to training without

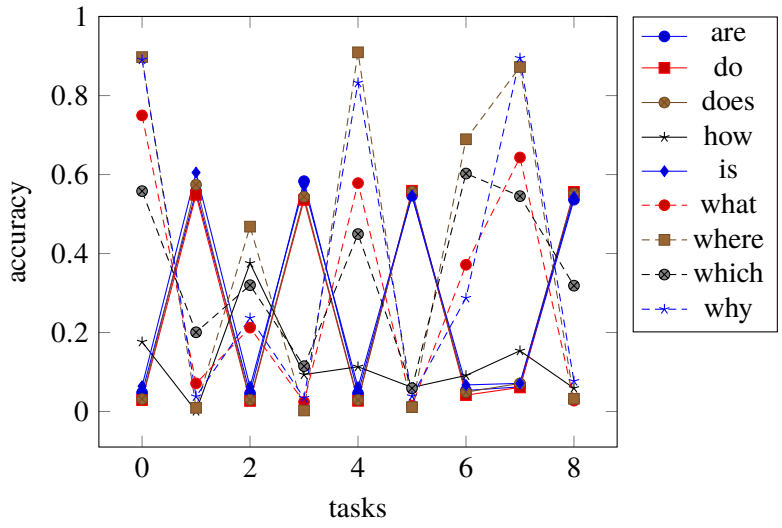

Figure 5.21: Accuracy of the VQA baseline with a reduced output to 50 classes for each question type.

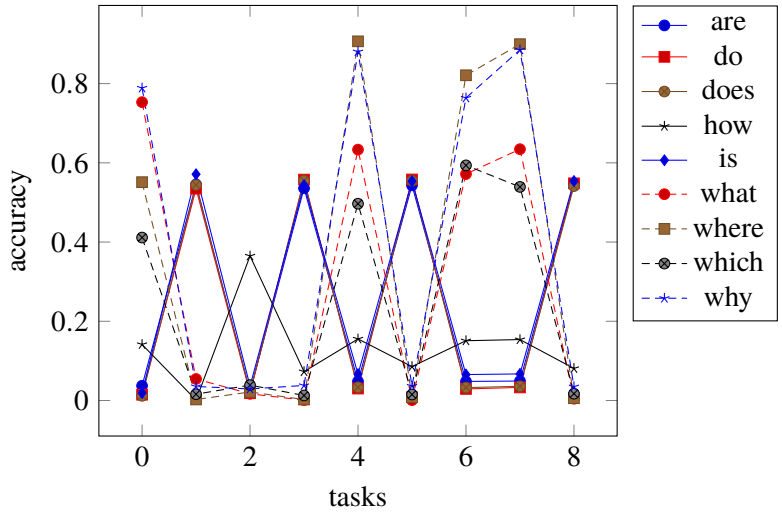

Figure 5.22: Accuracy of the VQA baseline with a reduced output to 50 classes. Question types are trained continually.

any continual learning method. When we look at Figure 5.23, we can see that the accumulated accuracies look almost the same. Training with and without generative replay makes no difference, the model's performance cannot be held across tasks.

It is possible that an accuracy of 65% is not high enough to avoid the error accumulation of creating labels for the generated feature vectors. The next step is to improve our model's accuracy even more to check if the error still accumulates and if this is actually the problem. However, another reason for the failure could be that our WGAN-GP is not capable of generating reasonable feature vectors for a complex dataset like the VQAv2 dataset.

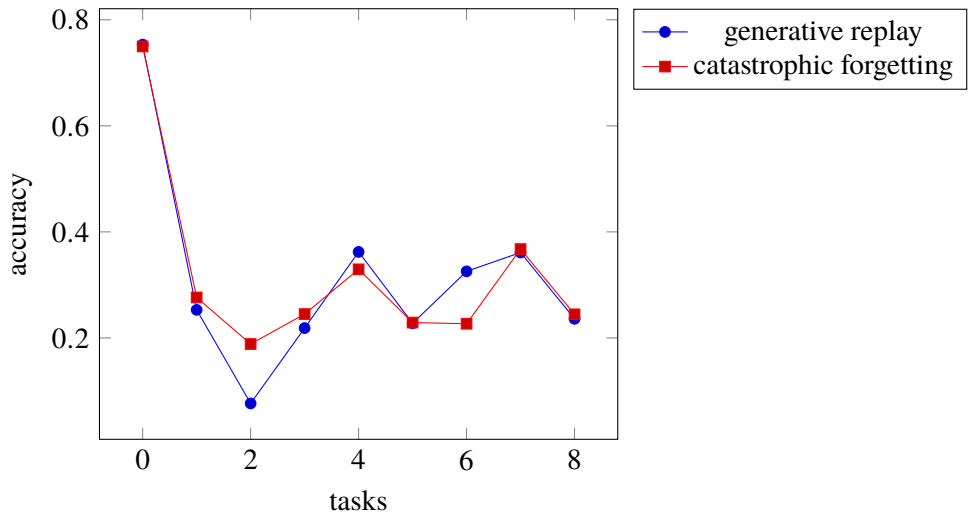

Figure 5.23: Accumulated accuracies of the VQA baseline with a reduced output to 50 classes. Trained with and without generative replay.

5.3.6 Only Correctly Classified Data

In this section, we train our model with only correctly classified data. We used our VQA baseline model to reevaluate which question and image pairs were correctly answered by our model. We trained our baseline model again with these data points only and expected to achieve a high accuracy.

Figure 5.24 shows the result of training only correctly classified data points. No generative replay was used. The accuracy is above 80% for 'what'-questions, which increases the performance of our baseline model. Now we can test, if a high accuracy of the task solver improves our results using generative replay as continual learning method. The hypothesis is that the error does not accumulate anymore. When tasks are trained without generative replay the accuracy drops again close to 0%. The consolidated graph displays testing on all validation datasets the model has seen so far. We reduced our testing to only two classes ('what'- and 'is'-questions), because we state that it is sufficient to train an open-ended questions type and a 'yes'/'no'-questions to show that continual learning with generative replay works (or not).

After showing catastrophic forgetting for our improved baseline model in terms of its accuracy of the question types, we can move on towards training it with generative replay and verify, if our hypothesis is correct. Figure 5.25 shows the result of training only correctly classified data. The accuracy of the "what"-question-type again decreases to almost 0% after training the "is"-question type. The curves of the "what"- and "is"-questions in Figure 5.25 and Figure 5.24 are very similar to each other. Table 5.4 summarizes the results of training only correctly classified data.

The experiment of increasing the accuracy of the baseline model to reduce the accumulated error when labeling the generated feature vectors was unsuccessful.

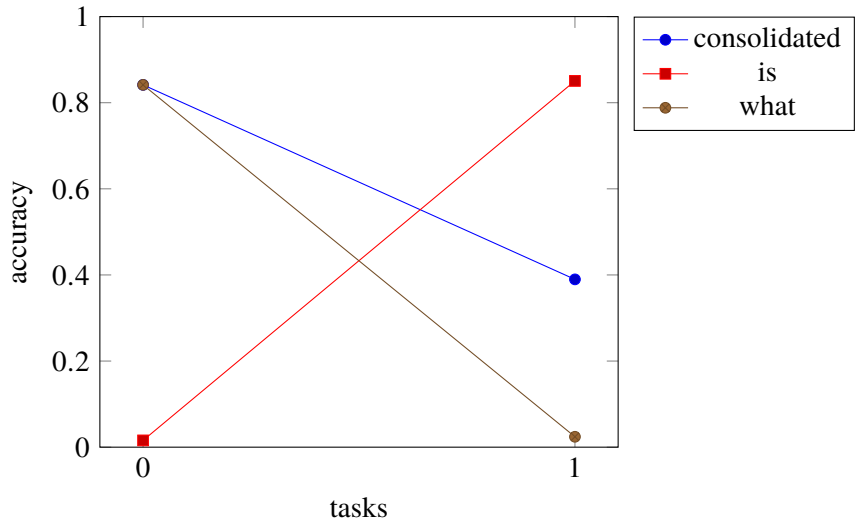

Figure 5.24: VQA baseline trained with only correctly classified question and image pairs. No generative replay was used.

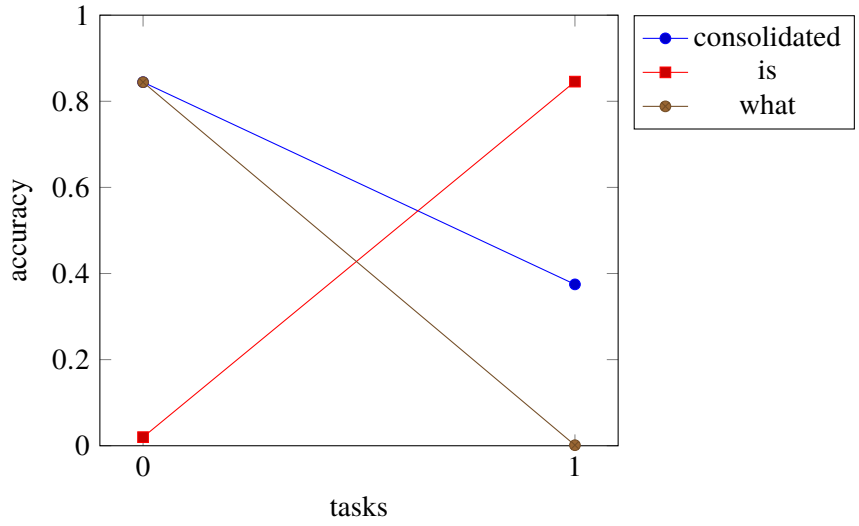

Figure 5.25: VQA baseline trained with only correctly classified question and image pairs. Generative replay was used.

5.3.7 Conclusion

In this chapter we integrated continual learning with generative replay in a VQA system. To reduce the complexity, we generated the combined feature vector of questions and images instead of generating both separately. After several approaches, we can state that it was unsuccessful to enable continual learning with our approach. The first, straight forward approach was to generate the feature vector of each question type and feed it into the network while learning a new question

Accuracy	Generative Replay	No Generative Replay
After first task	84.4%	84.1%
After all tasks	37.4%	38.9%

Table 5.4: Comparison of generative replay vs. no generative replay on the VQAv2 dataset with only correctly classified data.

type. Our experiment resulted in no memory improvement of previously learned tasks when our WGAN-GP was used to produce data samples. The accuracy of question types show that the VQA model still suffers from catastrophic forgetting.

As a next step, we decided to train only one question type continually and pre-train all other eight question types. The aim was to make it easier for our model to retain its already learned knowledge, due to a more excessive pre-training. The evaluation of the results again show that the memory capabilities of the baseline model do not improve. Using generative replay with an increased pre-training does not overcome catastrophic forgetting.

At this point, we narrowed the problem down to two reasons: the error of our wrongly labeled generated data accumulates or the generator is unable to capture and reproduce the feature vector space of images and questions. To exclude the first problem, we aimed to improve the initial accuracy of the VQA baseline with two different approaches: reducing the output to 50 classes and only use correctly classified data by the baseline.

Reducing the predicted output classes to 50 classes led to a performance increase of 17 percent points. The experiments to reduce catastrophic forgetting in our baseline failed again.

The second approach, to train the baseline only with correctly classified data, led to an accuracy of 84%. The error of labeling generated data samples wrong decreased and thereby the chance of fooling the baseline. The experiments on the other side stated that even this approach did not help to overcome catastrophic forgetting. We do not see any performance increase, when generative replay is used to enable continual learning for the VQA model.

Concluding, we state that the problem is not the accumulation of the error, but instead our WGAN-GP is unable to learn the data distribution of the VQAv2 dataset. Interpreting our findings and results leads us to the assumption that the variety and complexity within the VQA domain is too large to capture for a state-of-the-art GAN.

5.4 CLEVR

This section aims to reduce the complexity of the VQA domain and dataset. Johnson et al. [JHM+17] presented a diagnostic dataset, named CLEVR, to test a variety of visual reasoning abilities. CLEVR contains synthetic images with automatically generated questions. Every image contains objects of different shapes and colors. Shapes come in eight different colors and three different forms: cube, sphere and cylinder. The size is either small or large and the material can be shiny 'metal' or matte 'rubber'. The relationships between shapes can be 'left', 'right, 'behind' or 'in front' [JHM+17].

Questions are about the shape and color, for example how many objects exist of a specific color or shape inside the image. A neural network needs similar perceptual skills to extract the information out of the image and merge it with the information of the question.

Images of the CLEVR dataset only contain objects up to a fixed amount. As described, the objects get properties out of a defined set. A network gets trained on recognizing these objects with their relatively few properties. Compared to the VQAv2 dataset, this reduces the complexity by a large margin. CLEVR images contain less noise, due to their grey background. The background of real world images may contain objects itself. It thereby increases the noise for a neural network and makes it harder to focus on the relevant information. In VQAv2, images may contain all kinds of objects that are not relevant to the question and there is no limit for objects in real world images. These properties of real world images result in a higher complexity than CLEVR. Concluding we choose CLEVR for our next part of experiments.

5.4.1 Generating Images of CLEVR

We decided to return to generating images for the CLEVR dataset instead of generating feature vectors, because we believe that it is more promising for our WGAN-GP to capture the data distribution of this dataset than the data distribution of the VQAv2 dataset due to its higher variety. In the following, we review the results of generating images of size 128x128 and 64x64.

Figure 5.26 shows the results of generating images of size 128x128 with our WGAN-GP after training 240 epochs. The shapes of the object cannot be recognized. The background is too noisy

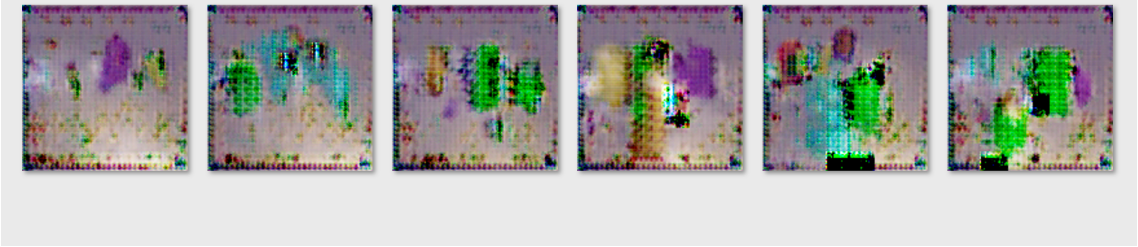

Figure 5.26: Examples of generated CLEVR images of size 128x128.

compared to the original images. The quality of the results is not promising enough to move to the next step and train a model continually. Instead we reduce the image size to 64x64 and verify, if our WGAN-GP can capture the distribution at this image size.

Figure 5.27 shows the results of generating images of size 64x64 with our WGAN-GP after training 240 epochs. The images seem to come closer to the original ones. The background color gets reproduced well, but the shapes and colors of the objects in the image itself are not of comparable quality. We state that the images are of a higher quality than the generated images of size 128x128, but still not good enough to move towards training a model with generated data samples of this quality.

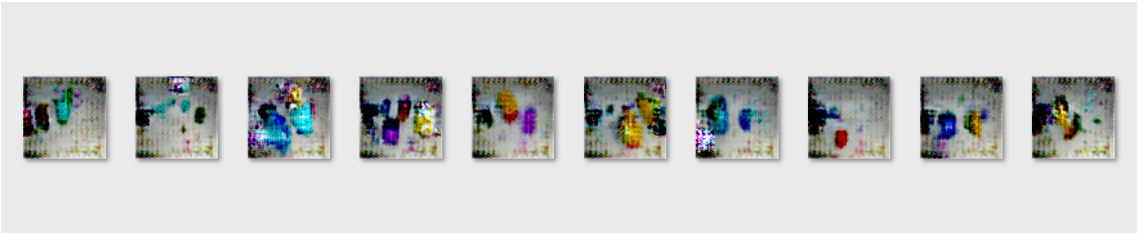

Figure 5.27: Examples of generated CLEVR images of size 64x64 after 500 epochs.

5.4.2 Conclusion

Our defined WGAN-GP was unable to generate images of the CLEVR dataset. Concluding, we did not implement a task solver for this dataset nor did we show catastrophic forgetting or try to enable continual learning.

Figure 5.28 shows generated images after 454 epochs of training on the dataset. Interestingly, at this

Figure 5.28: Examples of generated CLEVR images of size 64x64 after 454 epochs.

point, the WGAN-GP produced less noisy samples compared to the samples shown in Figure 5.27, although the loss of the critic was still decreasing. This confirms that the value of the loss function does not always indicate improved sample quality of the generator.

However, we decided to further decrease the complexity of datasets for the next step and implement our approach using the Shapeworld dataset.

5.5 Shapeworld

The poor sample quality and the concluding failure of using the CLEVR dataset as a less complex version for VQA led us to an even less complex VQA dataset, namely Shapeworld [KC17]. Kuhnle and Copestake [KC17] introduced Shapeworld for multi-modal deep learning models to evaluate their language understanding and generalization capabilities. The artificial data can be generated automatically during training or validation with respect to the experimenter's specifications. It contains abstract colored shapes, which are randomly sampled into the image at random positions. The language understanding part gets covered by generated captions for an image. The objective of the model is to agree or disagree with the generated captions by answering with yes or no. An important feature for us is the flexibility to specify how many shapes an image contains. Therefore, we can vary the complexity of an image while reducing the amount of shapes.

In Shapeworld, an image contains even less noise than images in the CLEVR dataset, due to the black background as well as the simple two-dimensional shapes, which can only vary in their positions and color. We can specify the images to only contain one shape at a time to further decrease the complexity. As a result, we have each image containing only one object that is placed on a black background. The language understanding part also gets reduced in their complexity by using simple captions for each image. The model only needs to agree or disagree with the caption by answering ”yes” or ”no”. We state that these properties of the Shapeworld dataset make it less complex compared to CLEVR. However, it still remains a multi-modal problem for VQA that requires a model to understand and combine the information of captions and images.

For this dataset, we define a continual learning task as follows: the model needs to learn to agree or disagree with captions and images of one specific shape. We randomly pick squares as the first task and triangles as the second shape to be learned. In other words, we feed the images of squares and corresponding captions into the network in the first iteration. In the second iteration, images of triangles and their captions are trained.

5.5.1 Generating Images of Shapeworld

The first step is to verify whether our WGAN-GP is able to generate images of Shapeworld. We start by generating images containing up to ten shapes in different colors.

Figure 5.29 shows the results of generating images containing up to ten shapes simultaneously. We trained the model for about 500 epochs containing 12,000 samples. The resulting images contain

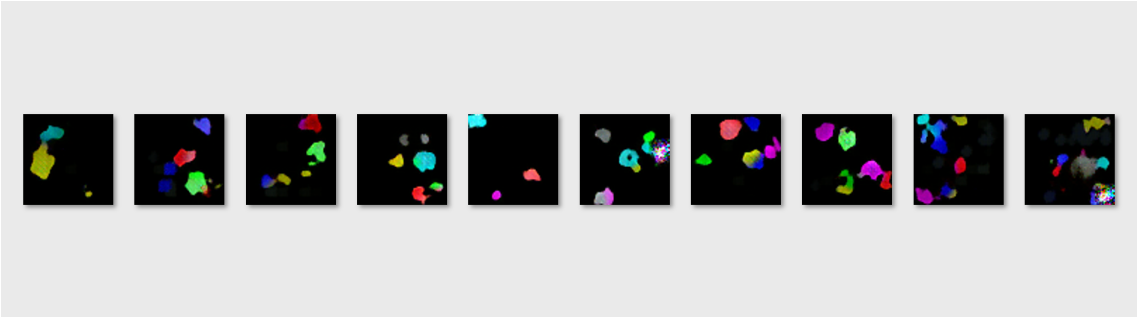

Figure 5.29: Generated Shapeworld images containing up to ten images.

figures that look blurry with no clear borders and shape. As a next step we reduce the amount of shapes to one and specify only one type of shape (in this case a square). Figure 5.30 displays the images of generating a square into the image. The edges still look blurry but the shapes can be identified. We state that the quality of the images is at least good enough to move on to the following step and use these generative samples as replays to enable continual learning.

5.5.2 Shapeworld Solver

Before we can show the results of continual learning in the domain of Shapeworld, we need to create a solver network that achieves a good performance on the Shapeworld dataset. Otherwise we cannot train continually. Kuhnle and Copestake [KC17] evaluated different types of models on

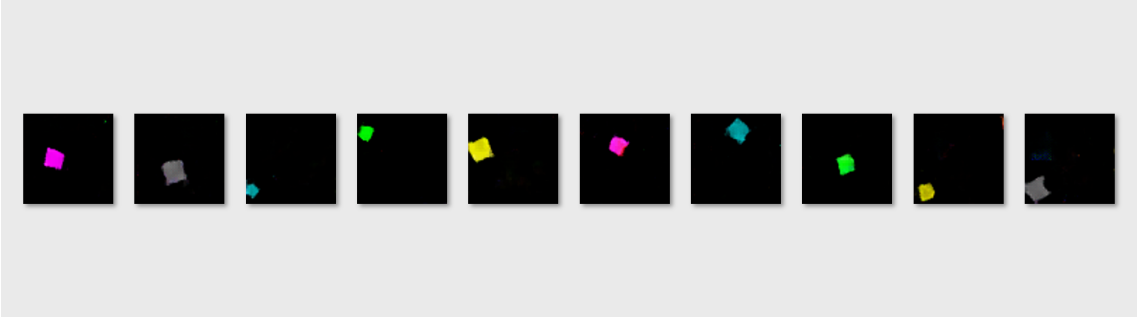

Figure 5.30: Generated Shapeworld images containing only a square.

Shapeworld. We decided to stick to the same type of architecture we used for the VQAv2 dataset: a CNN to extract the image features and a LSTM to extract the language features. The feature vectors also get combined with element-wise multiplication.

Due to the recurrent architecture we only describe the hyperparameters of our model. Table 5.5 shows the values of each hyperparameter we used. The output layer uses the sigmoid function as

Hyperparameter	Value
Learning Rate	0.001
Optimizer	Adam
Word Embedding	32
Output	1
LSTM Layers	2
LSTM Hidden Dimension	128
Dropout	0.5
Activation Function	ReLU

Table 5.5: Hyperparameters of our Shapeworld solver.

non-linearity.

Figure 5.31 shows the development of the accuracy of our defined Shapeworld solver. Kuhnle and Copestake [KC17] achieved an accuracy of 70% using a CNN+LSTM model with element-wise multiplication of the extracted feature vectors. Our model surpasses their result and achieves 75% on the validation data. A difference of our model compared to theirs is the use of dropout layers.

5.5.3 Catastrophic Forgetting in Shapeworld

In this section we show that our model suffers from catastrophic forgetting. We train images and corresponding captions of squares, followed by images containing triangles. Figure 5.32 displays the accuracy of the validation data of each task. We also evaluate the model with the combined validation data of both, squares and triangles, after each task was trained respectively. The accuracy of each task is higher than 90%, when it is trained with the original data. After a new task gets trained, the knowledge is overwritten, as we have seen for every domain within this

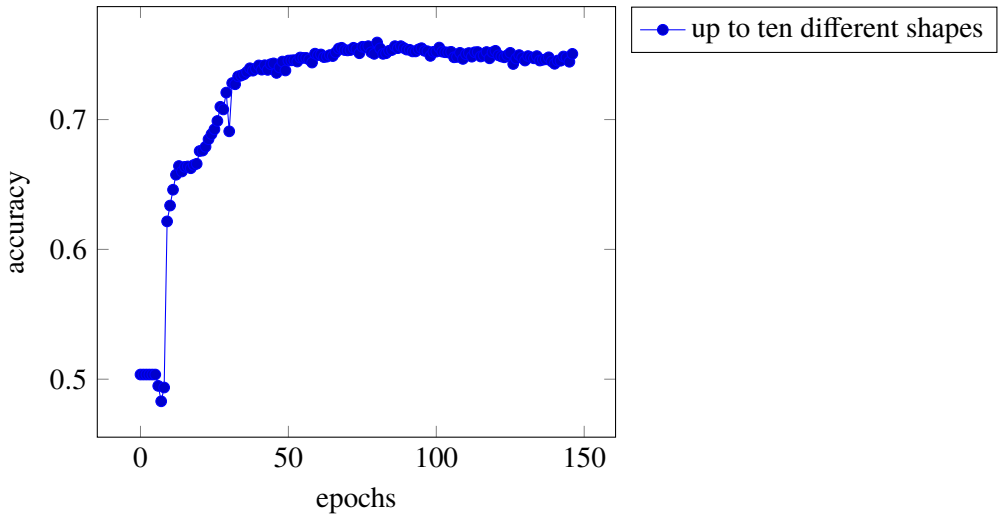

Figure 5.31: Accuracy of the Shapeworld solver trained on images containing up to ten different shapes.

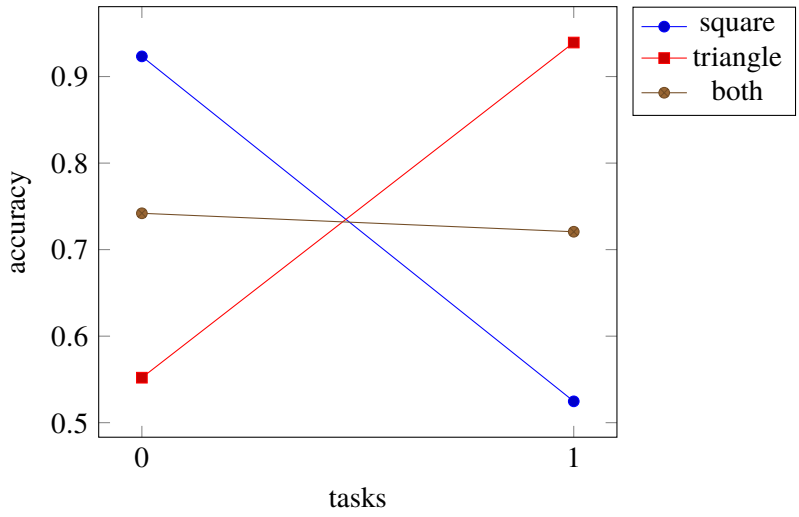

Figure 5.32: Catastrophic forgetting of our Shapeworld model.

thesis, and the network can only score a good performance on the most recently trained task. The performance of previously learned tasks drop to around 50%, what basically means answering questions arbitrarily.

5.5.4 Continual Learning in Shapeworld

After we showed that our model is unable to acquire new knowledge continually, we can now move on to use our generative replays to retain the information. We generate images using our defined WGAN-GP and decide to use a subset of questions from the dataset as input for the LSTM instead of

generating questions or question features. The reason behind the decision is the missing verification option if question features are meaningful or not. In the case of image generation humans can evaluate the result.

Algorithm 5.3 shows the training algorithm of our Shapeworld model, which slightly differs from MNIST and VQAv2. We iterate over different shapes s in all shapes $\mathcal{S}$. Due to the multi-modality,

Algorithm 5.3 Shapeworld training algorithm

```
for all s ∈ S do
    if s ≠ firstTask then
        generatedQstFeatures ← GANQuestions.sample
        generatedImages ← GANImages.sample
        labels ← ShapeworldSolver(generatedQstFeatures, generatedImages)
    end if
    ShapeworldSolver.train(s, generatedQstFeatures, generatedImages, labels)
    currentQstFeatures ← ShapeworldSolver.calculateFeatures(s)
    GANQuestions.train(currentQstFeatures, generatedQstFeatures)
    GANImages.train(currentQstFeatures, generatedImages)
    for all s ∈ S do
        ShapeworldSolver.test(s)
    end for
end for
```

we need two WGAN-GPs to generate the images as well as the questions or question features. We decided to generate question features to make it less complex compared to generating meaningful captions. After generating data samples for the generative replay phase while training a new task, the Shapeworld solver gets optimized with data from the new task. Generated data samples get spread into training the new data. Afterwards both WGAN-GPs get optimized to produce data samples from all tasks learned so far. To also produce data samples from previous tasks, we again use generated data samples to enable our GAN to also ”remember” to produce data from all tasks.

Figure 5.33 shows the results of using generative replays as continual learning method. This time we can see a clear performance boost when a new task is being trained. After training images containing triangles, the accuracy of agreeing or disagreeing with captions about images containing squares retains above 90%. Evaluating the model on the combined test datasets also shows an increased performance on both tasks. As a result, we can state that our approach of using generative replay as continual learning worked in the domain of Shapeworld with respect to our defined continual learning task.

Figure 5.34 shows the result of generating the question features vectors additional to the images. The performance of our solver on the data of squares drops close to 50%, meaning answering the captions arbitrarily. There is no difference to training the model without generative replay - the performance states that the model ”forgets” solving the previous task and results in catastrophic forgetting.

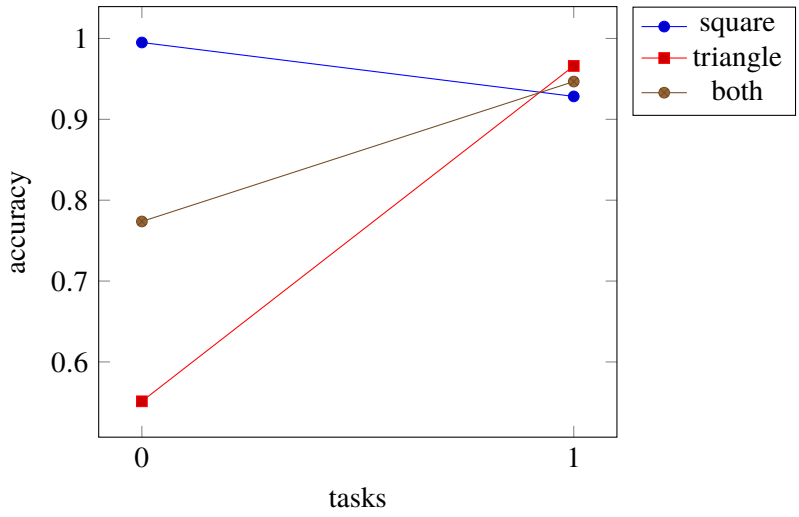

Figure 5.33: Continual learning with generative replay of our Shapeworld model with a batch of real questions.

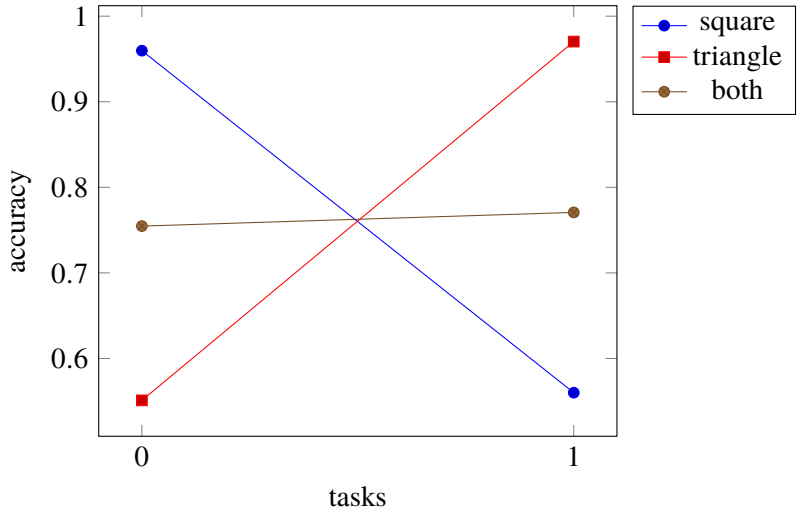

Figure 5.34: Continual learning with generative replay of our Shapeworld model with question features.

5.5.5 Conclusion

After a brief introduction into Shapeworld, we described our task solver, which achieved an accuracy higher than in the original paper. We showed that this model again suffers from catastrophic forgetting. The fact led us to implement our approach with generative replay for multi-modal dataset Shapeworld.

Table 5.6 summarizes the result of training images of squares and triangles sequentially. The

Accuracy	Generative Replay	No Generative Replay
Squares & Triangles after first task	77.3%	74.2%
Squares & Triangles after both tasks	94.6%	72.0%

Table 5.6: Comparison of generative replay vs. no generative replay on the Shapeworld dataset.

accuracy states that training with generative replays enabled continual learning and made it possible for the network to acquire knowledge of solving both tasks instead of only the most recent one.

Our first approach of generating images of Shapeworld paired with a batch of real questions from the previous task succeeded. We could retain the performance of the model on the previous data and therefore make the model ”remember”. We enabled continual learning for the multi-modal domain VQA using the dataset Shapeworld. Generating question feature vectors with a second generative model was unsuccessful and resulted in catastrophic forgetting.

6 Discussion

Continual learning can be applied through generative replays in the domain of VQA. But our findings suggest that it only works for simplified tasks, which do not contain too complex scenes. This might have several reasons, which we outline in the following paragraphs.

The performance of GANs are hard to evaluate objectively using any kind of metric. The loss functions of the critic and generator only give a hint as to how well the WGAN-GP is performing at the time. Generally, a scalar close to zero paired with only small changes of the loss between epochs determine that the WGAN-GP has converged. However, this does not always result in a better sample quality, as we have seen in Section 5.4. Although the loss was decreasing, the output images of the generator were more noisy than images from previous iterations. Feature vector generation is even harder to evaluate, because not even a human can determine if the generator is able to capture and reproduce the extracted high level features.

Beside the general problems with generative models, using them for continual learning is especially hard for multiple reasons. The success of continual learning relies heavily on the performance of the generator. If the generator is not able to reproduce the data distribution, the continual learning setting will fail. This becomes increasingly problematic in multi-modal setups where multiple generative models are needed to reproduce data samples from previous tasks to retain knowledge. Another problem is given through the error accumulation that can happen if the neural network, which is responsible for solving the actual task, only achieves a low performance. At replay time, the generated data samples get labeled falsely and the task solving network gets fed with wrong information and the continual learning setting fails.

The approach, we tried for the VQAv2 dataset and Shapeworld captions, to generate feature vectors instead of real data failed to enable continual learning in all our experiments. One possible reason could simply be the incapacity of our WGAN-GP to capture and generate the features. Another reason could be that it is not sufficient to slice the network to only feed in feature vectors. While training feature vectors, only the layers after the layer in which we feed in the feature vector, get updated accordingly. In the Shapeworld setting, the LSTM does not get updated to retain the information of the previous task, what could yield to failure of enabling continual learning. The same applies for the VQAv2 setting; only the last layers, the MLP, gets updated, the LSTM gets overwritten with the information of the new task.

However, generated samples do not need to be perfect in order to be good enough to enable continual learning with generative replay. As we have seen this for the Shapeworld dataset as well as the MNIST dataset that the data samples can still be distinguished from the real ones but they still worked to achieve a memory effect.

VQA is a complex, multi-modal domain. We evaluated three different datasets within this domain, two of them being synthetic. The variety in the VQAv2 dataset was too high for our WGAN-GP to capture the data distribution of the combined feature vector containing the information of images and

language. We think that there are too many things going on in a single image. Too much information for a state of the art GAN to capture and reproduce. To reduce the complexity we switched to a less complex VQA dataset named CLEVR. For us, the reduced complexity is demonstrated through less noise and less things going on in each image. Images of the VQAv2 dataset are real-world images that contain an overload of information. In contrast, the generated images of CLEVR are based on a grey background, which reduces the noise in the images. Each image only contains a fixed number of possible objects and these objects (with their shape, material and color) are repetitively used inside the dataset. Concluding, a neural network only needs to learn to recognize a relatively small number of objects with their varying properties. Comparing the fact to the VQAv2 dataset with real-world images, there is almost no limit of different objects the neural network can focus on. The variety of objects is therefore huge and most of them might not occur often enough for a neural network to learn to extract the relevant information needed to answer the corresponding question. At this point, we can derive a research question whether few-shot learning helps to enable continual learning for complex domains since only a network needs to extract knowledge from only a few samples.

However, our GAN still failed capture the data distribution and generate samples of any size of the CLEVR dataset. To focus on only one object at a time, we implemented our WGAN-GP for the Shapeworld dataset. Shapeworld tends to have even less noise than CLEVR - every image has a black background. Shapes in the dataset only vary in their color and position. As a result a model needs to learn fewer properties than for the CLEVR dataset. The model can agree or disagree with the captions by answering ”yes” and ”no”. The binary option of agreeing or disagreeing reduces the complexity of the task solver as well. The smaller the output of a model, the more likely continual learning with generative replay will succeed, because the error rate of falsely labeling generated data is reduced. This correlates with the performance of the task solver - lesser falsely labeled generated data samples are computed when the task solver reaches a high accuracy. Although the generated samples are not perfect, generative replays could enable continual learning and achieve a memory effect on previously learned tasks.

Our approach of using a set of questions and feeding them into the network paired with generated data might only work because in the Shapeworld domain, the network only needs to agree or disagree with the captions. Concluding every caption matches the image and makes sense to be asked. For more complex domains, storing a set of questions or captions might fail, because the image and the question have no relation to each other. There would be no suitable answer and continual learning would fail.

Lesort et al. [LCG+19] found comparable results as we did, but they did not investigate VQA. They state that generative models work particularly well on MNIST, but fail to perform good enough on more complex datasets like CIFAR10 to learn tasks continually. Awasthi and Sarawagi [AS19] stated the question whether the approach of generative replay can scale up to more complex domains, which we investigate in this thesis. We applied the approach on various VQA datasets. Our findings are that multi-modal problems like VQA require an immense image and language understanding and that generative replay does not perform well within the domain. Only the simplified Shapeworld dataset could be trained continually. Improvements within the field of generative models need to be made in order to capture more complex and varying data distributions to achieve a memory effect.

Generally, the idea of a comparable system to the complementary learning systems also works in neural networks. However, we could not observe a back flow of information while training new tasks which improve solving previously learned tasks. Our approach was only capable of

remembering already seen knowledge and retain almost the same performance as training it from scratch. Improvements regarding the generation capabilities need to be made in order to possibly improve generative replays. Training with higher sample quality and optimizing the amount of generated samples needed to retain specific knowledge could have a positive impact on increasing the performance of older tasks.

7 Conclusion and Future Work

In this thesis, we investigated generation-based continual learning methods for visual question answering. We started by explaining the origin and the background of our approach. The complementary learning system served as basic idea to enable continual or lifelong learning in neural networks. We covered continual learning methods, visual question answering in general and different kinds of deep learning techniques. We derived our approach from this information to experiment with the viability and effectiveness of generative models in continual learning settings on VQA datasets.

Our experiments showed that we could not overcome catastrophic forgetting with generating the combined feature vector of questions and images in the VQAv2 [GKS+17] setting. We compared the generated features to the real features by projecting into a two-dimensional space using PCA. To reduce the complexity of the VQA domain, we decided to move on to the CLEVR dataset. In this setting, we switched to generating images instead of feature vectors. This improved the possibility to evaluate the results, which are in this case images. A downside is that an objective evaluation is almost impossible and human evaluation is required. We did not implement a task solving model for CLEVR, because the results were not promising enough. Instead, we moved on to the next dataset and further decreased the complexity. Our approach did succeed in the Shapeworld setting when no question features were generated. We stored a set of questions that were fed into the network at generative replay time. The experiments failed when we tried to generate question features.

Many open questions remain when we want to enable continual learning for multi-modal problems, especially VQA. Improvements in the field of generation models will be needed to capture complex data distributions as VQA. Lesort et al. [LCG+19] also stated that GANs perform well on datasets like MNIST and SVHN, but produce samples of poor quality when trained on more complex datasets. GANs need to be able to capture a large variety in datasets with complex relations within and still produce a high sample quality. Furthermore, GANs require a large amount of computational time and resources to produce data samples of acceptable quality, which future advances need to reduce.

An additional research question, we want to address in the future, is to optimize the amount of generated replays needed for a task to keep its knowledge. This might differ from task to task and might possibly differ how many tasks have been trained previously. We also want to investigate the order tasks are getting trained in, although the order had no impact in our experiments so far. More evidence could reveal patterns inside the training order that positively affect continual learning.

Generating feature vectors was unsuccessful in all cases using our approach. Future investigations regarding this could improve the ability of easily enabling continual learning in various domains. Feature generation tends to be similar across datasets in any domain. Creating a generator to capture and reproduce feature vectors could lead to enable continual learning with minimized effort, because

it is not necessary to create a specific GAN that reproduces the input data. The investigation could uncover whether it is possible to enable continual learning through generative replays of feature vectors at all.

We did not investigate language generation models. An additional step in the future is to research the area of language generation models for continual learning. Although storing questions of previous tasks does not require much memory, it is desirable to generate pseudo-rehearsal data for language as well. We want to achieve the memory effect with generative replays in a multi-modal domain when multiple GANs are used.

Bibliography

[AAL+15] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, D. Parikh. "VQA: Visual Question Answering". In: *International Conference on Computer Vision (ICCV)*. 2015 (cit. on pp. 22, 33, 47).

[AB17] M. Arjovsky, L. Bottou. "Towards Principled Methods for Training Generative Adversarial Networks. arXiv e-prints, art". In: *arXiv preprint arXiv:1701.04862* (2017) (cit. on p. 29).

[ACB17] M. Arjovsky, S. Chintala, L. Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017) (cit. on pp. 18, 29, 30, 35, 39).

[AS19] A. Awasthi, S. Sarawagi. "Continual Learning with Neural Networks: A Review". In: *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. 2019, pp. 362–365 (cit. on pp. 20, 21, 34, 68).

[AW10] H. Abdi, L. J. Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459 (cit. on p. 31).

[BF16] M. K. Benna, S. Fusi. "Computational principles of synaptic memory consolidation". In: *Nature neuroscience* 19.12 (2016), p. 1697 (cit. on p. 19).

[BLS12] A. J. Bremner, D. J. Lewkowicz, C. Spence. *Multisensory development*. Oxford University Press, 2012 (cit. on p. 19).

[BSF94] Y. Bengio, P. Simard, P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166 (cit. on p. 27).

[DV16] V. Dumoulin, F. Visin. "A guide to convolution arithmetic for deep learning". In: *arXiv preprint arXiv:1603.07285* (2016) (cit. on p. 26).

[FBB+17] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, D. Wierstra. "Pathnet: Evolution channels gradient descent in super neural networks". In: *arXiv preprint arXiv:1701.08734* (2017) (cit. on p. 34).

[FG18] S. Farquhar, Y. Gal. "Towards robust evaluations of continual learning". In: *arXiv preprint arXiv:1805.09733* (2018) (cit. on p. 21).

[Fre99] R. M. French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135 (cit. on p. 19).

[GAA+17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. C. Courville. "Improved training of wasserstein gans". In: *Advances in neural information processing systems*. 2017, pp. 5767–5777 (cit. on pp. 18, 29, 30, 35, 39).

[GGHY15] D. Geman, S. Geman, N. Hallonquist, L. Younes. "Visual turing test for computer vision systems". In: *Proceedings of the National Academy of Sciences* 112.12 (2015), pp. 3618–3623 (cit. on p. 33).

[GKS+17] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, D. Parikh. “Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 22, 23, 33, 37, 39, 45, 46, 71).

[GMH+08] H. Gelbard-Sagiv, R. Mukamel, M. Harel, R. Malach, I. Fried. “Internally generated reactivation of single neurons in human hippocampus during free recall”. In: *Science* 322.5898 (2008), pp. 96–101 (cit. on pp. 17, 20).

[GPM+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. “Generative adversarial nets”. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on pp. 28, 29, 35).

[HS97] S. Hochreiter, J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 27).

[HVD15] G. Hinton, O. Vinyals, J. Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015) (cit. on p. 34).

[JHM+17] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, R. Girshick. “Clevr: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2901–2910 (cit. on pp. 33, 37, 58).

[KC17] A. Kuhnle, A. Copestake. “Shapeworld-a new test methodology for multimodal language understanding”. In: *arXiv preprint arXiv:1704.04517* (2017) (cit. on pp. 33, 37, 60–62).

[KGL17] N. Kamra, U. Gupta, Y. Liu. “Deep generative dual memory network for continual learning”. In: *arXiv preprint arXiv:1710.10368* (2017) (cit. on p. 34).

[KM12] D. Kumaran, J. L. McClelland. “Generalization through the recurrent interaction of episodic memories: a model of the hippocampal system.” In: *Psychological review* 119.3 (2012), p. 573 (cit. on p. 20).

[KMA+18] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, C. Kanan. “Measuring catastrophic forgetting in neural networks”. In: *Thirty-second AAAI conference on artificial intelligence*. 2018 (cit. on p. 34).

[KPR+17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526 (cit. on pp. 21, 34).

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 25).

[LBH15] Y. LeCun, Y. Bengio, G. Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 23–25, 27).

[LCB10] Y. LeCun, C. Cortes, C. Burges. “MNIST handwritten digit database”. In: (2010) (cit. on p. 34).

[LCG+19] T. Lesort, H. Caselles-Dupré, M. Garcia-Ortiz, A. Stoian, D. Filliat. "Generative models from the perspective of continual learning". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2019, pp. 1–8 (cit. on pp. 34, 35, 68, 71).

[LH17] Z. Li, D. Hoiem. "Learning without forgetting". In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947 (cit. on p. 34).

[LHB04] Y. LeCun, F. J. Huang, L. Bottou. "Learning methods for generic object recognition with invariance to pose and lighting". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 2. IEEE. 2004, pp. II–104 (cit. on p. 25).

[LMB+14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755 (cit. on p. 22).

[MF14] M. Malinowski, M. Fritz. "A multi-world approach to question answering about real-world scenes based on uncertain input". In: *Advances in neural information processing systems*. 2014, pp. 1682–1690 (cit. on p. 33).

[MMO95] J. L. McClelland, B. L. McNaughton, R. C. O'Reilly. "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." In: *Psychological review* 102.3 (1995), p. 419 (cit. on pp. 17, 19).

[MMPR19] M. Mundt, S. Majumder, I. Pliushch, V. Ramesh. "Unified Probabilistic Deep Continual Learning through Generative Replay and Open Set Recognition". In: *arXiv preprint arXiv:1905.12019* (2019) (cit. on p. 34).

[NWC+19] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Ng. *The street view house numbers (svhn) dataset*. 2019 (cit. on p. 34).

[OPDC10] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, J. Csicsvari. "Play it again: reactivation of waking experience and memory". In: *Trends in neurosciences* 33.5 (2010), pp. 220–229 (cit. on pp. 17, 20).

[PKP+19] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, S. Wermter. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* (2019) (cit. on pp. 17, 19, 20, 34).

[Rat90] R. Ratcliff. "Connectionist models of recognition memory: constraints imposed by learning and forgetting functions." In: *Psychological review* 97.2 (1990), p. 285 (cit. on p. 20).

[RHW85] D. E. Rumelhart, G. E. Hinton, R. J. Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985 (cit. on p. 25).

[RRD+16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016) (cit. on p. 34).

[SLKK17] H. Shin, J. K. Lee, J. Kim, J. Kim. "Continual learning with deep generative replay". In: *Advances in Neural Information Processing Systems*. 2017, pp. 2990–2999 (cit. on pp. 17, 34).

[SSB14] H. Sak, A. W. Senior, F. Beaufays. “Long short-term memory recurrent neural network architectures for large scale acoustic modeling”. In: (2014) (cit. on p. 26).

[SZ14] K. Simonyan, A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014) (cit. on p. 47).

[TSM15] K. S. Tai, R. Socher, C. D. Manning. “Improved semantic representations from tree-structured long short-term memory networks”. In: *arXiv preprint arXiv:1503.00075* (2015) (cit. on p. 28).

[TTK+11] D. Tse, T. Takeuchi, M. Kakeyama, Y. Kajii, H. Okuno, C. Tohyama, H. Bito, R. G. Morris. “Schema-dependent gene activation and memory encoding in neocortex”. In: *Science* 333.6044 (2011), pp. 891–895 (cit. on p. 20).

[VT19] G. M. van de Ven, A. S. Tolias. “Three scenarios for continual learning”. In: *arXiv preprint arXiv:1904.07734* (2019) (cit. on p. 22).

[WEG87] S. Wold, K. Esbensen, P. Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52 (cit. on p. 31).

[WHL+18] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al. “Memory replay gans: Learning to generate new categories without forgetting”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 5962–5972 (cit. on pp. 34, 35).

[YYLH17] J. Yoon, E. Yang, J. Lee, S. J. Hwang. “Lifelong learning with dynamically expandable networks”. In: *arXiv preprint arXiv:1708.01547* (2017) (cit. on p. 34).

[ZGS+16] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, D. Parikh. “Yin and Yang: Balancing and Answering Binary Visual Questions”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 22, 33).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature