COMENIUS UNIVERSITY, BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

GENERATIVE PROPERTIES OF BIO-PLAUSIBLE NEURAL MODEL UBAL

MASTER'S THESIS

2023 BC. ZUZANA HALGAŠOVÁ

COMENIUS UNIVERSITY, BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

GENERATIVE PROPERTIES OF BIO-PLAUSIBLE NEURAL MODEL UBAL

MASTER'S THESIS

Study program:	Cognitive Science
Study field:	Computer Science
Department:	Department of Computer Science
Supervisor:	RNDr. Kristína Malinovská, PhD.

Bratislava, 2023

Bc. Zuzana Halgašová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Zuzana Halgašová
Študijný program: kognitívna veda (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Generative properties of bio-plausible neural model UBAL
Generatívne vlastnosti biologicky plauzibilného neurálneho modelu UBAL

Anotácia: Najčastejšie používané pravidlo riadeného učenia v umelých neurónových sieťach, spätné šírenie chyby (BP) [1], sa považuje za biologicky neplauzibilné. Model UBAL (Universal Bidirectional Activation-based Learning) je biologicky vierohodnou alternatívou k BP, ktorá nepoužíva chybové gradienty. Emergentnou vlastnosťou modelu UBAL je, že generuje obrazy dát, ktoré sa učí klasifikovať, bez toho, aby bol na to trénovaný [2]. Tzv. adverzariálne príklady [3] sú obrázky navrhnuté tak, aby oklamali natrénované neurónové siete, vytvorené pomocou šumu a gradientov chýb siete. Predpokladáme, že keďže model UBAL nepoužíva chybové gradienty, je robustný voči takýmto adverzariálnym príkladom.

Cieľ: Dôkladne preskúmajte model UBAL a experimentálne vyhodnotte generatívnych vlastností modelu s trénovaného na benchmark datasete MNIST a vyhodnoťte kvalitu vygenerovaných vzorov. Posúďte odolnosť modelu UBAL voči nepriaznivým vzorkám.

Literatúra: [1] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. (1986). Learning Internal Representations by Error Propagation". In Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol. 1. MIT Press.
[2] Malinovská, K., Farkaš, I., 2021, Generative Properties of Universal Bidirectional Activation-Based Learning. ICANN 2021. Springer, Cham.
[3] Goodfellow, I.J., Shlens, J. and Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv:1412.6572.

Vedúci: RNDr. Kristína Malinovská, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 01.10.2021

Dátum schválenia: 01.03.2022 prof. Ing. Igor Farkaš, Dr.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Zuzana Halgašová
Study programme:	Cognitive Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Generative properties of bio-plausible neural model UBAL

Annotation: The most popular supervised learning rule in artificial neural networks, the error backpropagation (BP) [1], is considered biologically implausible. Universal Bidirectional Activation-based Learning (UBAL) is a bio-plausible alternative to BP, not using error gradients. An emergent property of UBAL is that it generates patterns it learns to classify, without being trained to do so [2]. The adversarial examples [3] are images designed to fool trained neural networks, created using noise or the error gradients from the network. We hypothesize that projections of UBAL can be used to create adversarial examples.

Aim: Study the UBAL model and make thorough experimental evaluation of the generative properties of the model with a benchmark dataset (MNIST) and the influence of the noise added to the labels. Evaluate the generated patterns in terms of their quality. Assess the possibility of using the images generated by UBAL as adversarial samples using selected networks.

Literature:
[1] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. (1986). Learning Internal Representations by Error Propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1. MIT Press.
[2] Malinovská, K., Farkaš, I., 2021, Generative Properties of Universal Bidirectional Activation-Based Learning. ICANN 2021. Springer, Cham.
[3] Goodfellow, I.J., Shlens, J. and Szegedy, C., 2014. Explaining and harnessing adversarial examples. arXiv:1412.6572.

Supervisor:	RNDr. Kristína Malinovská, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	01.10.2021
Approved:	01.03.2022

prof. Ing. Igor Farkaš, Dr.
Guarantor of Study Programme

Student

Supervisor

Pod'akovanie: I want to thank several people who have supported me throughout my studies. Firstly, I am grateful to my supervisor Kristina Malinovska for providing guidance. I am also thankful to my friends, Katka and Ondrej, for taking care of me throughout my studies. Additionally, I would like to thank my parents for their unwavering support. Finally, a special thank you to my girlfriend for her patience, cooking, and keeping an eye on me while I completed my studies.

Abstrakt

Táto práca skúma UBAL, nový biologicky vierohodný model umelej neurónovej siete (ANN) so zaujímavými vlastnosťami pri spätnom šírení. UBAL využíva dve sady váh pre priame a spätné šírenie a ukázal sa ako sľubný pri generovaní obrazov počas spätného šírenia. To z neho robí potenciálne výkonný nástroj na generatívne úlohy. Cieľom tejto štúdie je lepšie pochopiť UBAL a preskúmať jeho odolnosť voči adverzariálnym obrazom. Výkonnosť a generatívne schopnosti UBAL sa hodnotili prostredníctvom experimentov s použitím súboru údajov MNIST a adverzariálnych obrazov generovaných útokom FGSM.

Kľúčové slová: umelé neurónové siete, UBAL, adverzariálne obrazy, adverzariálny útok, MNIST

Abstract

This thesis explores UBAL, a novel biologically plausible Artificial Neural Network (ANN) model with intriguing properties in backward propagation. UBAL utilizes two sets of weights for forward and backward propagation and has shown promise in generating images during backward propagation. This makes it a potentially powerful tool for generative tasks. The study aims to understand UBAL better and explore its robustness against adversarial images, UBAL's performance and generative abilities were evaluated through experiments using the MNIST dataset and adversarial images generated by the FGSM attack.

Keywords: artificial neural networks, UBAL, adversarial images, adversarial attack, MNIST dataset

Contents

List of Figures

List of Tables

Acronyms

ANN artificial neural networks

BAL Bidirectional Activation-based Learning algorithm

BP Error back-propagation

CHL Contrastvie Hebbian Learning

EP Equilibrium Propagation

FGSM Fast Gradient Sign Attack

FID Fréchet Inception Distance

GAN Generative Adversarial Network

GeneRec Generalized Recirculation

IS Inception Score

UBAL Universal Bidirectional Activation-based Learning

VAE Variational Autoencoder

Introduction

Artificial Neural Networks (ANNs) have been instrumental in advancing various domains, demonstrating remarkable capabilities in tasks such as image classification and recognition. However, traditional ANNs can not completely capture the complexity and biological plausibility of the human brain's learning mechanisms. This thesis explores a novel biologically plausible ANN model called Universal Bidirectional Activation-based Learning (UBAL), which exhibits intriguing properties.

UBAL differentiates itself by utilizing two sets of weights for forward and backward propagation, mimicking the asymmetric nature of neural connections in biological systems. UBAL has an emergent ability to generate images during backward activation propagation, making it an intriguing candidate for generative tasks. The quality of these generated images is influenced by the hyperparameters of the learning rule, opening possibilities for further research.

We focus on investigating its robustness towards adversarial images – carefully crafted inputs designed to deceive and mislead conventional ANNs. The ability of UBAL to withstand such attacks would have significant implications for enhancing the reliability and security of image classification and recognition systems.

We employ the widely used MNIST dataset and adversarial images generated using the Fast Gradient Sign Method (FGSM) attack to evaluate UBAL's performance. Our experimental approach involves assessing UBAL's mean accuracy per epsilon and mean accuracy per label per epsilon, and constructing a confusion matrix based on the classification outcomes. Additionally, we explore the generative properties of UBAL by utilizing standard MNIST images as input data and evaluating the quality of images created by UBAL using the Fréchet Inception Distance (FID) metric.

The thesis unfolds in a logical progression, beginning with a brief overview of how learning occurs in the brain, contrasting different types of learning in ANNs, and highlighting the limitations of the widely adopted error-backpropagation learning algorithm in terms of biological plausibility. We then delve into biologically plausible alternatives, including Generalized Recirculation (GeneRec) and UBAL's predecessor, Bidirectional Activation-based Learning algorithm (BAL). Subsequently, we provide a short explanation of UBAL's architecture, it's biologically plausible learning, and its unique generative capabilities. We also discuss modern bio-inspired models and their relevance to UBAL. Moreover, we intro-

duce the concept of Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), shedding light on adversarial images and their creation process. Finally, we outline the experimental setup and methodology adopted to investigate UBAL's performance and generative abilities.

This thesis contributes to the growing knowledge of biologically plausible ANN models, focusing on UBAL's characteristics, robustness towards adversarial images, and generative capabilities.

1 Theoretical introduction

In order to comprehend the uniqueness of the UBAL model, we must delve into various topics. These include learning in the brain, the learning algorithm commonly used by ANNs, and why it is biologically implausible. Additionally, to fully grasp the experiments in this thesis, we need to review several concepts. Thus, this chapter is divided into several sections: biological mechanisms of learning, ANNs and their different types of learning, why the most prominent learning algorithm is biologically implausible, UBAL and its unique qualities, and lastly, sections explaining concepts necessary for understanding the experiments chapter.

1.1 Learning in the brain

The alteration of synaptic weights in a neural network due to the local activity of the sending and receiving neurons is what is referred to as learning. This alternation is called synaptic plasticity. As these synaptic weights define what each neuron detects, they are essential to predicting the behavior of individual neurons and entire networks. To put it another way, all of our knowledge is stored in the patterns of our synaptic weights, which all our experiences have formed. There are two primary types of learning - self-organizing and error-driven:

- **Self-organizing learning**, gathers more extended time-scale statistics about the environment and can thus help develop an effective internal model of the outside world (i.e., what kinds of things tend to happen in the world reliably) [1].
- **Error-driven learning** employs more rapid contrasts between expectations and results to adjust these expectations and thus acquire more particular, in-depth knowledge about world contingencies [1].

Self-organizing learning involves averaging over a long time scale, e.g., what happens when we blur our eyes and take stuff in over some time. On the other hand, error-driven learning is much faster; it requires much more alert and rapid forms of neural activity [1]. The most effective learning is a combination of these two methods.

1.1.1 Synaptic plasticity

Synaptic plasticity (learning) involves modifying the effectiveness of a synapse that connects two neurons. It is a *local mechanism* that adjusts the strength of individual connections based on local activity patterns. There are many moving elements in the synapse, and any of them might be the decisive factor in changing its effectiveness. The early research stage on synaptic plasticity was dominated by the search for vital factors. Evidence for the involvement of numerous different aspects has been uncovered over time. The number of presynaptic neurotransmitters released, the number and effectiveness of postsynaptic *AMPA receptors*, the alignment of pre and postsynaptic components, and the cloning of multiple synapses. The number and effectiveness of postsynaptic AMPA receptors appear to be the primary factor for long-lasting learning alterations [1].

Several crucial stages are driving the alteration of AMPA receptors efficacy. The calcium ion (Ca++) and the *NMDA receptors* are vital because NMDA channels allow Ca++ to reach the postsynaptic spine. Ca++ usually plays a significant part in controlling cellular function in all body cells. In the neuron, it can trigger a chain of chemical events that ultimately regulates the number of functioning AMPA receptors in the synapse. Since these AMPA receptors provide the neuron's primary excitatory input drive, altering them alters the overall excitatory effect of a presynaptic action potential on the postsynaptic neuron. This is what we refer to as changing the synaptic weight [1].

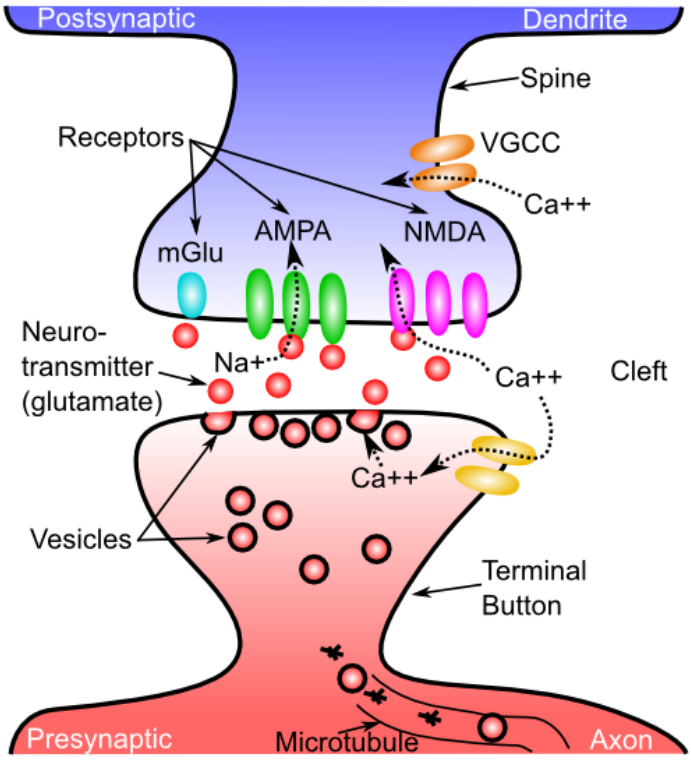

Figure 1.1: Schematic of a synapse, showing presynaptic neuron releasing neurotransmitter into the synaptic cleft [1].

AMPA receptors efficacy may change in two directions - increase or decrease. The biological term for long-lasting efficacy increases is Long Term Potentiation (LTP); long-lasting decreases are called Long Term Depression (LTD). This change direction depends on the overall level of Ca++ in the postsynaptic spine – low levels drive LTD, while high levels produce LTP as can be seen in Figure 1.2 [1].

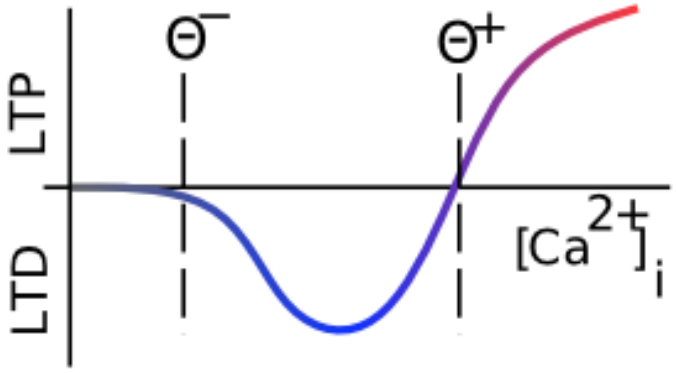

Figure 1.2: Direction of synaptic plasticity (LTP = increase, LTD = decrease) as a function of Ca++ concentration in the postsynaptic spine [1].

Hebbian Learning

Psychologist Donald O. Hebb is famously known for his quote regarding learning in the brain:

Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability... When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased [8].

Which can be summed up as *cells that fire together, wire together*.

Mathematically, it can be represented as:

$$\Delta w = xy \tag{1.1}$$

where Δw is the change in synaptic weight w, x is sending activity and y is receiving activity.

1.2 Introduction to Artificial Neural Networks

Artificial neural networks (ANNs) are a subset of learning algorithms inspired by biological neural networks. They are a tool for simulating the human brain, so in the end, the computer will be capable of learning and making decisions like a human brain. Similarly, as the human brain, they consist of interconnected nodes, also known as neurons. These neurons work together to process and learn from the data. A typical artificial neural networks (ANN) consists of three types of layers: an input layer, one or more hidden layers, and an output layer. Data is fed into the network through the input layer, and the network processes the data through the hidden layers before producing a final output through the output layer. Simply put, ANN are functions. Based on input, they provide an output.

Three basic learning algorithms in ANN are supervised learning (also known as error-driven), unsupervised learning, and reinforcement learning. In further sections, each learning algorithm will be briefly explained, with an example of the most used training algorithm of ANN, why this model is not biologically plausible, and several alternatives.

1.2.1 Supervised learning

An ANN is trained using labeled data in supervised learning. The ANN learns to make predictions by being shown examples of inputs and corresponding outputs. The goal is for the network to understand the relationship between the inputs and outputs to predict the outcome for new inputs accurately that it has not seen before. Supervised learning in ANN is commonly used in applications such as image recognition, speech recognition, and natural language processing [9].

1.2.2 Unsupervised learning

Unsupervised learning is a type of learning where the algorithm learns patterns and relationships in the data without explicit labels or target outputs. In other words, the algorithm is given a set of inputs and tries to find hidden structures and relationships within the data. Unlike supervised learning, where the algorithm is given labeled data to learn from, unsupervised learning works with unlabeled data [9].

1.2.3 Reinforcement learning

Reinforcement learning deals with how agents can learn to make optimal decisions based on feedback from their environment. It involves training a neural network to make decisions that maximize a reward given to the agent based on its actions. The algorithm consists of three main components: the agent, the environment, and the reward. The agent is the neural network, which receives input from the environment and outputs an action. The environment

is the world in which the agent operates, and it provides feedback to the agent in the form of a reward. The reward is a value that indicates how good or bad the agent's action was [10].

1.3 Error Back-propagation

The most prominent supervised learning algorithm has been the Error back-propagation (BP) since its origin in the 1980s when it was created by David Rumelhart and others in 1986 [11]. Its goal is to adjust the weights of the connections between neurons in the network so that the network can make more accurate predictions on a given task. The basic idea of BP is to calculate the error between the predicted output of the network and the actual output and then use that error to adjust the weights of the connections between the neurons in the network. The algorithm works by propagating the error backwards through the network, from the output layer to the input layer, and using that error to calculate the gradient of the loss function concerning the weights.

Two phases make up the conventional error BP process, forward pass phase and backward pass phase. In the forward pass, the network receives its initial input as activation of the input layers neurons, which spreads throughout the network and generates an estimate on the output layer. The error is then propagated backward through the weights of the network as the difference between the desired and estimated values on the output layer (backward pass) [11].

In a classical multilayer perceptron (MLP) with input layer x, output layer y and one hidden layer h, using error backpropagation rule, weights are updated according to

$$\Delta W_{hy} = \lambda \delta_y h, \text{ where } \delta_y = (d-y)f', \tag{1.2}$$

and

$$\Delta W_{xh} = \lambda \delta_h x, \text{ where } \delta_h = (W_{hy}^{\mathsf{T}} \delta_y)f', \tag{1.3}$$

where $\lambda > 0$ is the learning rate, y is the network's output and d is the desired value to be learned. , W_{hy} is the weight matrix connecting the output and the hidden layer, W_{xh} connects input and hidden layer, f is the activation function, in original form it is the sigmoid [11]:

$$\sigma(\eta) = 1/(1+\exp(-\eta)). \tag{1.4}$$

1.3.1 Biological plausibility

Relatively soon after its inception, Francis Crick [12] and Stephen Grossberg [13] questioned BP's biological implausibility. Grossberg brought up the weight transport issue [13]. Since each neuron must know all of its feed-forward connections to update its weights based on the error signal from the output layer, the weights in BP are transported throughout the network. Meaning BP requires a *global error signal*. Nevertheless, connections in the brain are "unaware" of the synaptic potency of the neurons upstream or downstream of them - learning happens through local mechanisms (see section 1.1). The rapid information transfer demanded by this method, according to Crick [12], is also incompatible with the actual

axonal transmission capabilities of neurons. He also pointed out that the brain often does not function linearly but rather on a local level.

Several models were proposed to solve BP's implausibility, including the algorithm developed by Mazzoni et al. based on reinforcement learning [14]. Unfortunately, these models were primarily created to fit certain data types (e.g., brain activity data from monkeys); therefore, they were not suggested as general-purpose learning algorithms for ANNs. On the other hand, the GeneRec model by O'Reilly [15] was proposed directly as a biologically plausible alternative to error back-propagation and has been tested on canonical tasks such as the 4–2–4 encoder and XOR problem.

1.4 Bio-plausible alternatives to Error Back-propagation

1.4.1 GeneRec

As mentioned in the previous section (refer to 1.3), BP is biologically implausible because it requires the error propagation mechanism, and it does not use locally available, activation-based variables. O'Reilly [15] created the Generalized Recirculation (GeneRec) algorithm, keeping this in mind; it does not require the computation of error derivatives but can still result in error minimization. GeneRec was created to expand Rumelhart and McClelland's autoassociation-only model[16], which relied on recirculation between two layers of units (visible and hidden) with symmetric weights. They employed a four-stage activation update technique to make it function. On the other hand, GeneRec adopts a two-phase activation, as depicted in the figure 1.3 update process, and is implemented in a fully connected three-layer network with bidirectional interaction between the hidden and output layer (see table1.1 for activation rules). The two activation phases are:

- In the **forward phase**, or minus phase (-), the outputs represent the network's *expectation* as a function of the standard activation process in response to a particular input. It produces the network's *estimate* of the output/target values as described in [2].
- In the **backward** or **plus phase** (+), the desired output is clamped on the output layer of the network and then propagated in the opposite direction, using the same set of weights as in the forward direction [2].

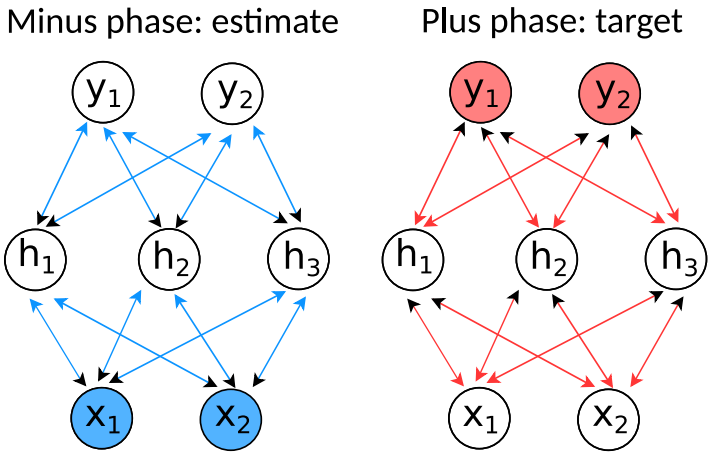

Figure 1.3: Illustration of the plus and the minus phase in GeneRec. [2]

The learning rule is applied to both input-hidden and hidden-output weights:

$$\Delta W_{pq} = \lambda \; p^{-}(q^{+} - q^{-}), \tag{1.5}$$

where p^- represents the presynaptic and q^- represents the postsynaptic activation in the minus phase, p^+ is the presynaptic activation from the plus phase. λ denotes the learning rate [2].

Table 1.1: Activation propagation rules between connected layers in GeneRec.

Layer	Ph.	Net Input	Activation
In(s)	–	-	s_i = stimulus inp
Hid(h)	–	$\eta_j^- = \sum_i w_{ij} s_i + \sum_k w_{kj} o_k^-$	$h_j^- = \sigma(\eta_j^-)$
	+	$\eta_j^+ = \sum_i w_{ij} s_i + \sum_k w_{kj} o_k^+$	$h_j^+ = \sigma(\eta_j^+)$
Out(o)	–	$\eta_k^- = \sum_j w_{jk} h_j$	$o_k^- = \sigma(\eta_k^-)$
	+	-	o_k^+ = target out

1.4.2 Bidirectional Activation-based Learning

BAL is similar to GeneRec regarding phase-based activations and unit kinds. However, it generates completely bidirectional connections (GeneRec focuses on input-to-output mapping). In contrast to GeneRec, BAL employs two pairs of weight matrices for both forward (F) and backward (B) activation propagation (as illustrated in 1.4) [2].

The fundamental difference between the GeneRec model and the BAL model is that instead of a typical input-output design, the network's visible layers are both inputs and targets, and connectivity is fully bidirectional. As a result, the layer notation in this model differs since the output might be elicited by the presentation of the input and vice versa. The visible layers are labeled x and y, the hidden layer is labeled h [2].

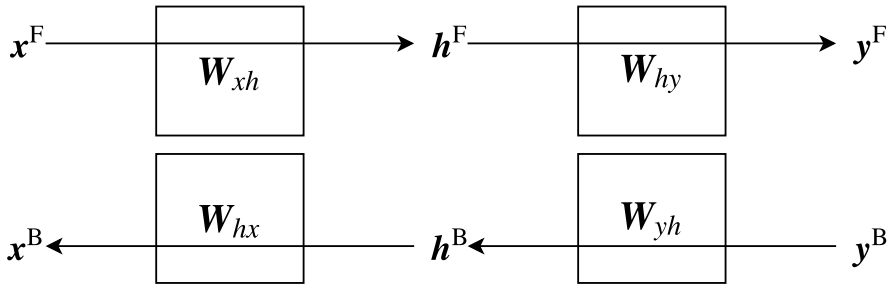

Figure 1.4: Illustration of a three-layer BAL network.

The learning rule adjusts the weights depending on local changes in the forward (F) and backward (B) activation phases. The weights in the forward direction (p to q) for two connected layers, p and q, are updated as follows:

$$\Delta W_{pq} = \lambda \; p^F (q^B - q^F), \tag{1.6}$$

and analogically weights in backward direction (q to p) are updated as

$$\Delta W_{qp} = \lambda \; q^B (p^F - p^B), \tag{1.7}$$

where λ is the learning rate [2].

The experimental evaluation of the BAL algorithm demonstrated that it could create hetero-associative mappings between pairs of sparse random binary patterns [2] and performed similarly to GeneRec in the 4-2-4 encoder task. BAL, on the other hand, was unable to converge on the XOR task, unlike the original GeneRec model. After that, Csiba and Farkaš [17] analyzed the BAL algorithm and created BAL2, in which λ could be set for each weight matrix separately. Nevertheless, even the best solution did not perform on the 4-2-4 encoder task as well as GeneRec. In order to overcome these difficulties, UBAL model was proposed.

1.5 Universal Bidirectional Activation-based Learning

As mentioned in the previous section, a UBAL [3] algorithm was proposed to overcome the shortcomings of the BAL model. Not only it converges in the 4–2–4 encoder task, but unlike its predecessors, it also converges in XOR and other classification tasks. Although UBAL was initially designed for bidirectional hetero-associative mappings, it can also be adapted to perform unidirectional many-to-one associations, such as classification. With the help of additional learning parameters, it can even generalize over a class of instances.

1.5.1 Activation flow and variables in UBAL

UBAL, like BAL, creates bidirectional connections between pairs of data patterns. The network is designed for hetero-associative mapping, meaning the inputs are also targets and vice versa, rather than just input and output. The visible layers of the network are referred to as x and y. The activation flow has a forward and backward direction, corresponding to the $x-y$ and $y-x$ associations, and is propagated through separate weight matrices for each direction. We label the weights W for the *forward* direction (F) and M for the *backward* direction (B) [3].

This algorithm, also its predecessor BAL, has a unique feature not found in GeneRec: it considers weight asymmetry, directly solving the weight transport problem [13]. Other models, such as the Feedback Alignment family, address this problem using separate feedback connections. Its bidirectionality requires training weights M along with weights W to form meaningful connections in both directions, controlled by specific hyperparameters [3].

Similar to GeneRec and other models, UBAL goes through various activation stages. In this model, the activation during the plus phase corresponds to the minus phase activation in the opposite direction. To prevent confusion, we will refer to the typical activation flow from the input on the visible layer to the network's prediction on the other visible layer (from the layer x to y) as the *prediction phase (P)* instead of using the terms **plus** and **minus** phase [3].

In contrast to the previous BAL model, a new feature is introduced that shares similarities with the regression mechanism proposed by Hinton and McClelland [18]. This mechanism involves the activation of the postsynaptic layer being sent back to the presynaptic layer (from the layer y to x) after each prediction phase, referred to as the *echo phase (E)* [3]. As seen in the figure 1.5, if activation occurs on a hidden layer, we can propagate it back through the M weights. This applies to both visible and hidden layers. The architecture in the figure is simple, containing only one fully connected layer.

The UBAL network is designed to be scalable, accommodating varying numbers of layers. Table 1.2 displays the general activation propagation rule between two connected layers p and q, where f is the activation function and b and d are trainable biases with constant input 1.0. For our present project, we utilize the conventional Sigmoid activation function

(Eq. 1.4). However, it's worth noting that alternative activation functions can also be employed, and these may vary across different layers of the model [3].

Table 1.2: Activation propagation rule between connected layers p and q.

Step	Direction and phase	Label	Activation
1.	Forward Prediction	FP	$q^{\mathrm{FP}} = f(W_{pq}p^{\mathrm{FP}} + b_p)$
2.	Forward Echo	FE	$p^{\mathrm{FE}} = f(M_{qp}q^{\mathrm{FP}} + d_q)$
3.	Backward Prediction	BP	$p^{\mathrm{BP}} = f(M_{pq}q^{\mathrm{BP}} + d_q)$
4.	Backward Echo	BE	$q^{\mathrm{BE}} = f(W_{qp}p^{\mathrm{BP}} + b_p)$

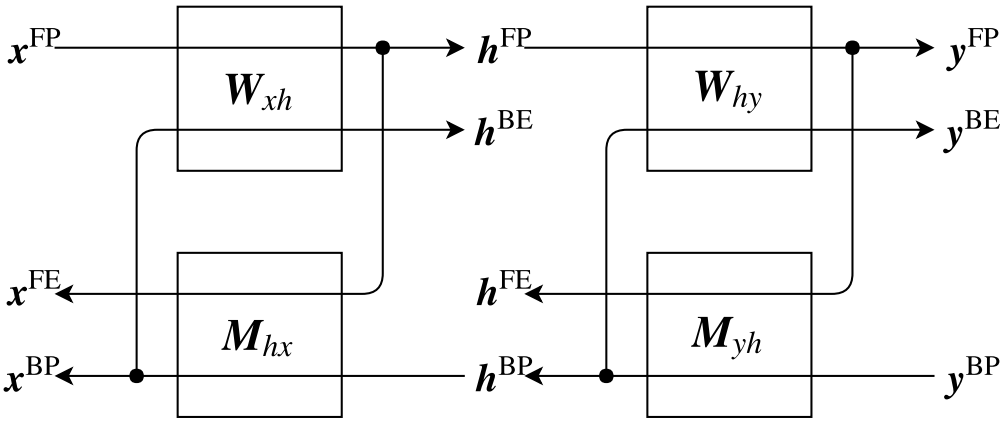

Figure 1.5: Scheme of a three layer UBAL [3].

We consider a simple architecture of one fully connected layer. The activation propagation scheme in such a network can be seen in Fig. 1.5.

1.5.2 Learning in UBAL

The weight update rule in GeneRec, BAL, and similar models is based on the contrastive Hebbian-anti-Hebbian principle. This means that the weight update is proportional to the product of the input term and error terms' products. The error term is the difference between the desired activation and the network's actual estimate. The input term can either come from propagating the input to the next layer or from clamping the input value on the visible layer. In a general form, the input term refers to the presynaptic layer activation [3].

When dealing with UBAL, the input and error terms become more complex. It must take into account the prediction, training signal, and internally recirculated echo states. UBAL also controls the significance of these two network states in terms of network learning. The table 1.3 shows the intermediate learning rule terms t and e, which are part of the general learning rule described below. Meanwhile, the Fig. 1.6 illustrates the activation states of the network [3].

Table 1.3: Learning rule terms

Term	Symbol	Value
Forward Target	t_q^{F}	$\beta_q^{\mathrm{F}} q^{\mathrm{FP}} + (1-\beta_q^{\mathrm{F}}) q^{\mathrm{BP}}$
Forward Estimate	e_q^{F}	$\gamma_q^{\mathrm{F}} q^{\mathrm{FP}} + (1-\gamma_q^{\mathrm{F}}) q^{\mathrm{BE}}$
Backward Target	t_p^{B}	$\beta_p^{\mathrm{B}} p^{\mathrm{BP}} + (1-\beta_p^{\mathrm{B}}) p^{\mathrm{FP}}$
Backward Estimate	e_p^{B}	$\gamma_p^{\mathrm{B}} p^{\mathrm{BP}} + (1-\gamma_p^{\mathrm{B}}) p^{\mathrm{FE}}$

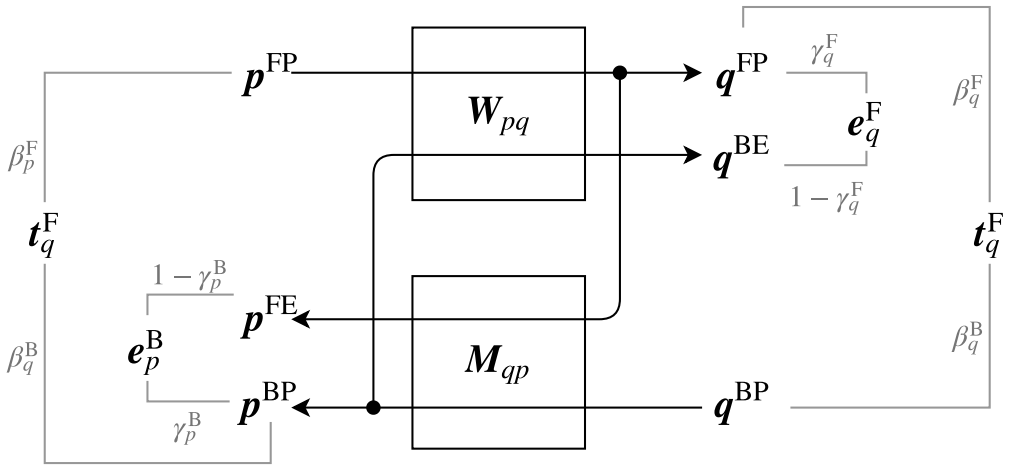

Figure 1.6: Activation propagation and learning rule terms for connected layers p and q in UBAL with intermediate terms and hyperparameters [3].

The weights between two layers in the forward direction in the learning algorithm are updated according to

$$\Delta W_{pq} = \lambda \; t_p^{\mathrm{B}}(t_q^{\mathrm{F}} - e_q^{\mathrm{F}}) \tag{1.8}$$

and in the backward direction according to:

$$\Delta M_{qp} = \lambda \; t_q^{\mathrm{F}}(t_p^{\mathrm{B}} - e_p^{\mathrm{B}}) \tag{1.9}$$

The β and γ hyperparameters control the proportions of the intermediate terms, which are crucial for network learning. By adjusting these values, UBAL can effectively learn and solve various problems [3].

1.5.3 Learning rule hyperparameters

Learning in the UBAL model is influenced by special hyperparameters β and γ that are used in the learning rule terms (refer to Table 1.3). The β hyperparameter allows for the so called weak clamping (also used in related models described in Sec. 1.6), i.e. the proportion of activation from the forward and backward pass that enters the weight update formula as target. Since UBAL uses the input signal in one direction (F or B) as a teaching signal for the opposite direction, the target term in each layer is the same for both directions of association. By using the equation $t_n^{\mathrm{F}} = t_n^{\mathrm{B}}$ for layer n, we can determine that $\beta_n^{\mathrm{B}} = (1-\beta_n^{\mathrm{F}})$ for all layers.

The second special hyperparameter introduced in the UBAL model is denoted by γ. γ^F and γ^B are used in the learning rule to determine the strength of *prediction* in the given direction compared to the *echo* from the other direction when creating the estimated term by mixing results given by the updated weights. Unlike βs, γs cannot be reduced to just one direction as they are not related to a layer of neurons but instead to a weight matrix connecting the two layers. When using extreme values $(0, 1)$ of β and γ, the components of learning rule reduce to different forms of learning that resemble canonical algorithms such as CHL. Thus, UBAL can perform different kinds of learning (supervised, unsupervised, semi-supervised).

1.5.4 Generative properties of UBAL

UBAL's performance in the MNIST benchmark is similar to related models when using the hyperparameter setups listed in Table 1.4 and a hidden layer with at least 1500 neurons. UBAL can achieve up to 96% accuracy on the testing set without any supplemental optimization techniques. To achieve this, Malinovská and Farkaš [4] used a 3-layer network with standard sigmoidal units (Softmax for output layer), Gaussian weight initialization N(0.0,0.5), and a learning rate of 0.05. One-hot vectors were used to encode the MNIST digit targets, and images were normalized to (0, 1) [4].

Table 1.4: Two setups of UBAL hyperparameters for MNIST [4].

	Setup A	Setup B
β^{F}	0.0 - 1.0 - 0.0	1.0 - 1.0 - 0.9
γ^{F}	1.0 - 1.0	1.0 - 1.0
γ^{B}	1.0 - 1.0	0.9 - 1.0
β^{B}	1.0 - 0.0 - 1.0	0.0 - 0.0 - 1.0

Since UBAL is a hetero-encoder, apart from classifying the digits, it also makes projections of those digits in its input layer. These projections can be considered the network's imagination, as depicted in Fig. 1.7 and are distinct from the computed averages of all images in the dataset (Fig. 1.8. The initial findings indicate the ability of creating meaningful images depends on the hyperparameters setup. Table 1.4 shows two different setups that achieve similar performance, yet produce different patterns 1.9. By reducing the hidden layer β^{F} from 1.0 to a lower value (0.995 - 0.999999), the accuracy may slightly decrease, but the resulting images have softer edges and more variability [4]. Furthermore, we will call them backprojections.

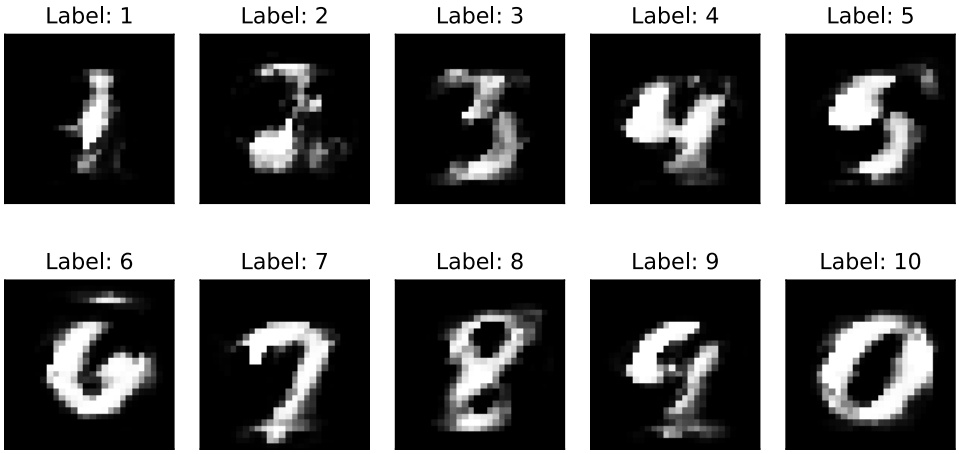

Figure 1.7: UBAL backprojections of all 10 digits. [4]

Figure 1.8: Projection of computed mean of all MNIST samples for the 10 digits.

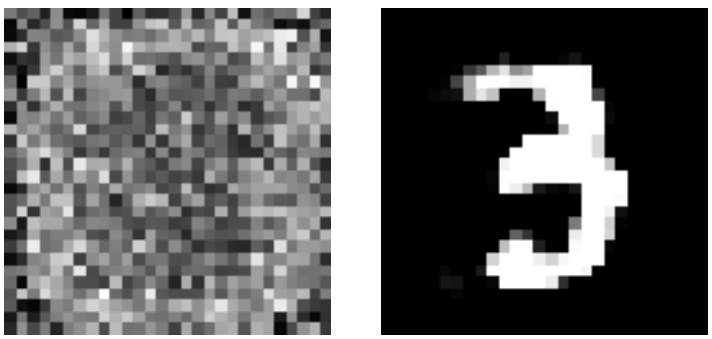

Figure 1.9: Example of digit 3 with Setup A (left) and Setup B (right) [4].

1.6 Modern bio-inspired models

Biologically plausible learning is gaining popularity, especially in its potential link to Deep Learning, as the current state-of-the-art models continue to depend on standard BP learning.

1.6.1 Error backpropagation alternatives

Xie and Seung [19] found that Contrastive Hebbian learning can achieve error backpropagation in certain cases. Scellier and Bengio [20] built on this research by exploring whether BP can be implemented in an energy-based model. They proposed the Equilibrium Propagation (EP) algorithm, which works similarly to Contrastvie Hebbian Learning (CHL) and GeneRec by settling into an equilibrium point with each input propagation. EP also learns based on the difference in activation values in different activation phases. However, it allows for "weak clamping," where the activation of clamped neurons is only partially driven towards the desired activation state, with the proportion being parameterized.

O'Reilly suggested a new learning rule called the eXtended Contrastive Attractor Learning (XCAL)[21] in his Leabra framework [22], building upon the previous GeneRec model. This rule is specifically designed for more complex spiking neuron models with greater biological detail. By implementing the plus phase activation clamping principle, XCAL is able to facilitate unsupervised and supervised learning based on the duration of the presynaptic neuron firing activity.

Like O'Reilly's models, Lillicrap's feedback alignment (FA) [23] model draws significant inspiration from neuroscience. However, Lillicrap's approach differs from the criticism of the BP model by avoiding the propagation of activation through the same set of weights. Despite still using standard gradient descent learning, the FA model uses random feedback weights B to deliver teaching signals instead of $W^\intercal$, making it a δ-rule model. Lillicrap argues that functional symmetry is sufficient, meaning that the random matrix B only needs to match the shape and properties of $W^\intercal$ to function similarly.

1.7 Generative models

Artificial intelligence has made great strides thanks to Generative Neural Networks. These networks, such as Generative Adversarial Network (GAN) and Variational Autoencoder (VAE), have been particularly noteworthy for their ability to learn and reproduce complex data patterns. With GANs and VAEs, we can generate new and realistic samples, leading to exciting developments in AI creativity, image creation, and anomaly detection. This section will explore the foundational concepts behind GANs and VAEs.

1.7.1 Generative Adversarial Networks

Generative Adversarial Networks are a class of machine learning models introduced by [5]. GANs have revolutionized the field of generative modeling by enabling the creation of realistic synthetic data, such as images, music [24], and text [25], that closely resemble real-world examples.

GANs operate by utilizing two neural networks called the generator and the discriminator. The generator learns to produce synthetic data by converting random noise samples from a latent space to the data space. Meanwhile, the discriminator learns to identify the differences between the original and generated data. The process of training GANs is a two-player minimax game between the generator and discriminator networks. They are trained iteratively, with the generator trying to create samples that can trick the discriminator. On the other hand, the discriminator aims to classify accurately whether a sample is original or generated—leading to a feedback loop that drives both networks to improve. Initially, the generator creates random samples while the discriminator is taught to classify real data correctly. As training progresses, the generator learns to generate more realistic samples that are increasingly difficult for the discriminator to distinguish from real data. Simultaneously, the discriminator improves its ability to distinguish real samples from the generated ones [5].

Since the original publication of the GAN paper in 2014, there have been significant advancements and variations in GAN technology, which have expanded their applications and capabilities. These include conditional GANs [26], which enable generated samples to be controlled by specific inputs, and progressive GANs [27], which produce images with increasing resolution.

1.7.2 Variance autoencoders

Variance autoencoders, or variational autoencoders (VAEs), are a type of neural network that integrates elements of both generative models and autoencoders. Autoencoders are neural networks that aim to learn efficient representations of input data through compression into a lower-dimensional latent space, then reconstruct the original input from this compressed

Figure 1.10: Here are some examples of data created by Generative Adversarial Networks. The networks were trained using two datasets, MNIST and Toronto Faces Dataset (TFD). In each set of examples, the rightmost column shows real data closest to the generated data in the neighboring columns. This demonstrates that the network creates new data rather than just copying what it has already seen. [5]

representation. However, traditional autoencoders are limited as they cannot generate new data samples.

On the contrary, variance autoencoders overcome this limitation by using a probabilistic approach to create data. They add a probabilistic layer to the encoder-decoder framework, which helps the model learn the probability distribution of the input data. This enables VAEs to encode and decode data, as well as create new samples from the learned distribution [28].

Variance autoencoders work by modeling the latent space as a multivariate Gaussian distribution. To achieve this, the network includes two additional components: the encoder network, which maps input data to the latent space distribution, and the decoder network, which generates data samples from the latent space [28].

When being trained, VAEs have two main goals: minimizing reconstruction loss and Kullback-Leibler (KL) divergence. The reconstruction loss measures how different the original input is from its reconstructed version, which helps the VAE learn how to encode and decode data well. The KL divergence objective aims to make the latent space distribution that the VAE learns to match a standard Gaussian distribution as closely as possible. This objective encourages the VAE to learn how to represent the latent space smoothly and continuously [28].

VAEs optimize these two objectives to encode input data into the latent space. Each point in the latent space corresponds to a probability distribution. This representation allows for sampling new data points by randomly selecting latent variables and decoding them back into the original data space [28].

1.8 MNIST Dataset

The MNIST dataset is a popular benchmark in machine learning, especially for image recognition and classification. It is called Modified National Institute of Standards and Technology, named after the organization that developed the original dataset. The dataset is a collection of handwritten digits presented as 28x28-pixel images. It comprises a training set of 60,000 examples and a test set of 10,000 examples. Each image in the dataset represents a digit from 0 to 9. The goal is to develop a machine-learning model that accurately identifies the digit in each image. The dataset is preprocessed and organized to make it convenient for training and evaluating machine learning models. The images are grayscale, containing only shades of gray, with pixel values ranging from 0 to 255. The labels associated with each image specify the correct digit it represents [29].

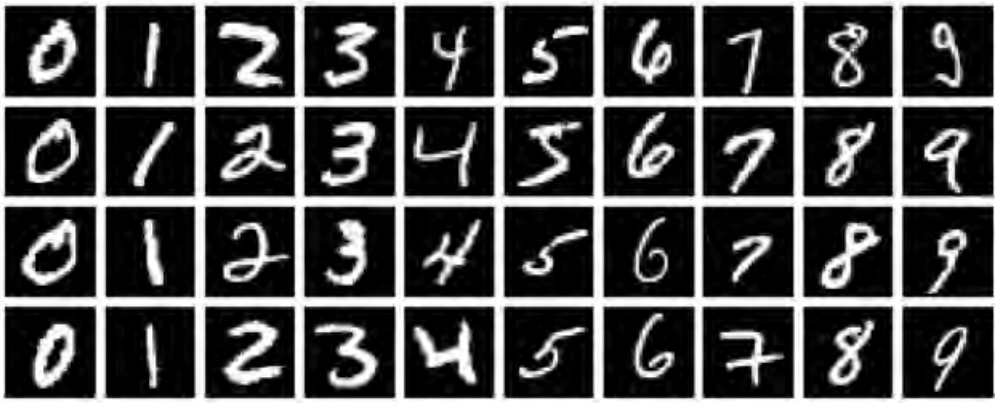

Figure 1.11: Sample images from the MNIST handwritten digits database [6].

1.9 Adversarial Images

Adversarial images are phenomena in which minor, carefully made perturbations to an input image induce a well-trained neural network to misclassify the image. These disturbances are frequently unnoticeable to the human eye, yet they can be highly effective in causing a neural network to give the wrong output [7].

These perturbed images are created by adversarial attacks that can be classified into categories based on the attacker's goals and assumed knowledge. The main objective is to cause a misclassification by adding minimal perturbation, ε, to the input data. The attacker's knowledge can either be *white-box* or *black-box*, depending on their access to the model's architecture, inputs, outputs, and weights. In a *white-box* attack, the attacker has full access to the model, while in a *black-box* attack, they only have access to the inputs and outputs. The types of goals include *misclassification* and *source/target misclassification*. *Misclassification* aims to produce incorrect output classification without specifying the new classification. *Source/target misclassification*, on the other hand, aims to change the image's classification from a specific source class to a specific target class. Goodfellow described the Fast Gradient Sign Attack (FGSM), one of the earliest and most widely used adversarial attacks. It is a white-box attack with the goal of misclassification.

An example of an adversarial image can be seen in Figure 1.12 below [7].

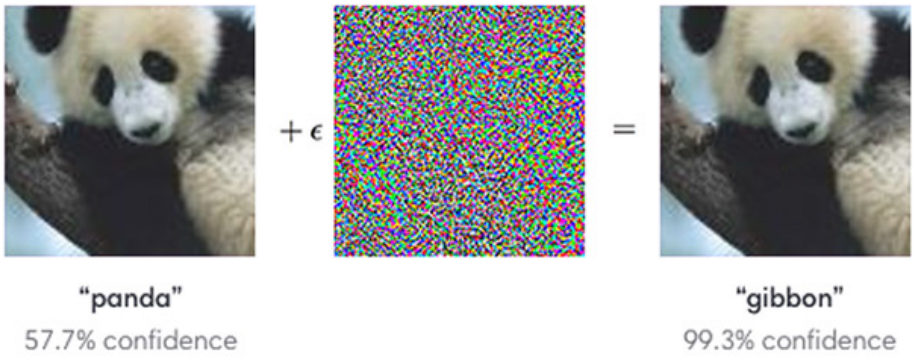

Figure 1.12: Example of a fooling image attack on a neural network. The image on the left is the original image, and the image on the right is the adversarial image created by adding the ε, noise, to the original image. The neural network misclassifies the adversarial image as a "gibbon" instead of a "panda" [7].

The impact of adversarial images can be severe, especially in applications where the neural network's output directly impacts human lives. In autonomous vehicles, for example, an attacker could use adversarial images to force the vehicle to misidentify a stop sign as a different one, perhaps leading to an accident.

1.10 Inception Score and Fréchet Inception Distance

In the domain of neural networks, Inception Score (IS) and Fréchet Inception Distance (FID) are commonly used metrics for evaluating generative models. These metrics are important in generative models because they help researchers assess the quality and diversity of generated images. By comparing and analyzing different models, researchers can use these metrics to gain insight into the effectiveness of the models they are working with.

The Inception Score, introduced by Salimans in 2016 [30], is a metric for measuring the quality and diversity of generated images. This is achieved using the Inceptionv3 network and leveraging its pre-training on a large dataset like ImageNet to evaluate the generated samples. The Inception Score comprises two components: the average of the predicted class probabilities for each generated image, which indicates how well the images can be classified into meaningful categories. The second component is the entropy of the predicted class probabilities, which captures the diversity of the generated samples. The Inception Score provides a single scalar value that combines these two factors to quantify the quality and diversity of the generated images.

Although the Inception Score is commonly used, it has some drawbacks. It tends to favor models that produce images recognizable to the pre-trained Inceptionv3 network but may not capture the diversity of real-world images. Moreover, it does not consider the generated images' distribution and cannot capture similarities or differences in the overall structure of the generated and real data [31].

In 2017, Heusel [31] created the Fréchet Inception Distance (FID) to address the limitations of the IS. The FID uses the Inceptionv3 network to compare real and generated images based on their distribution rather than individual samples. Specifically, it measures the distance between the multivariate Gaussian distributions of the feature embeddings obtained from the Inceptionv3 network for the real and generated images. The difference between the two Gaussians (real-world images and generated) is measured by the Fréchet distance [32], also known as the Wasserstein-2 distance [33]. The Fréchet distance $d(.,.)$ between the Gaussian with mean (m,C) and the Gaussian with mean (m_w,C_w), is what they called the *Fréchet Inception Distance*, which is given by [34]:

$$d^2((m,C),(m_w,C_w)) = \|m-m_w\|_2^2 + Tr(C+C_w-2(CC_w)^{1/2}) \tag{1.10}$$

It measures the quality and diversity of generated images and considers the data's structure and distribution. A lower FID score means that the distributions of real and generated images are similar, indicating that the generated samples are of higher quality and diversity [31].

For our thesis, we have chosen to utilize the FID as our preferred method.

2 Experiments

In earlier chapters, we discussed the BP algorithm 1.3. We explained why it is not biologically plausible—also introducing various alternatives, including the new neural network model, UBAL. As mentioned previously, UBAL has a unique feature where it can create projections of digits in its input layer without being specifically designed for image generation like GANs and VAEs.

This chapter is dedicated to exploring this innovative model through various experiments. The experiments aimed to assess UBAL's accuracy when classifying the MNIST dataset, its robustness against adversarial images, and also assess images generated by UBAL with different γs and βs.

Based on preliminary results [4], hyperparameter β_3 influences the resulting images created by UBAL. Therefore we decided to use several trained network models with different β_3 hyperparameters. The adversarial images were generated by an already existing neural network model [35]. We will refer to this model as the **generator** further on.

In our experiments, we utilized four different models in three layered UBAL that varied in terms of the number of neurons in each hidden layer, learning rate, β and γ hyperparameters, and the number of epochs. Further details are provided in Table 2.1. For clarity, we named the models **A** , **B** , **C** , and **D** . β and γ values are taken from the table 1.4, we labeled β_3 because it is the most important difference in the models (setups).

Table 2.1: Details of the models

Name	Number	Hidden layers	Learning rate	β_3	Epochs
Model **A**	1618785365	1500	0.05	0.0	50
Model **B**	1618786994	1500	0.05	0.9	60
Model **C**	1684862409	1800	0.1	0.9	80
Model **D**	1684862638	1800	0.1	0.9	120

We utilized the Pytorch Library [36] for all the experiments. As a result, we created reusable scripts tailored for Python's UBAL code attached to this thesis (see Appendix).

2.1 UBAL's accuracy in the MNIST dataset

We used 10,000 images from the MNIST testing dataset as input for the five models. Our analysis examined the accuracy of each model and which labels were most commonly misclassified. We will provide bar charts that show overall accuracy, accuracy per label for each model, and confusion matrices that indicate which numbers were mistaken for which.

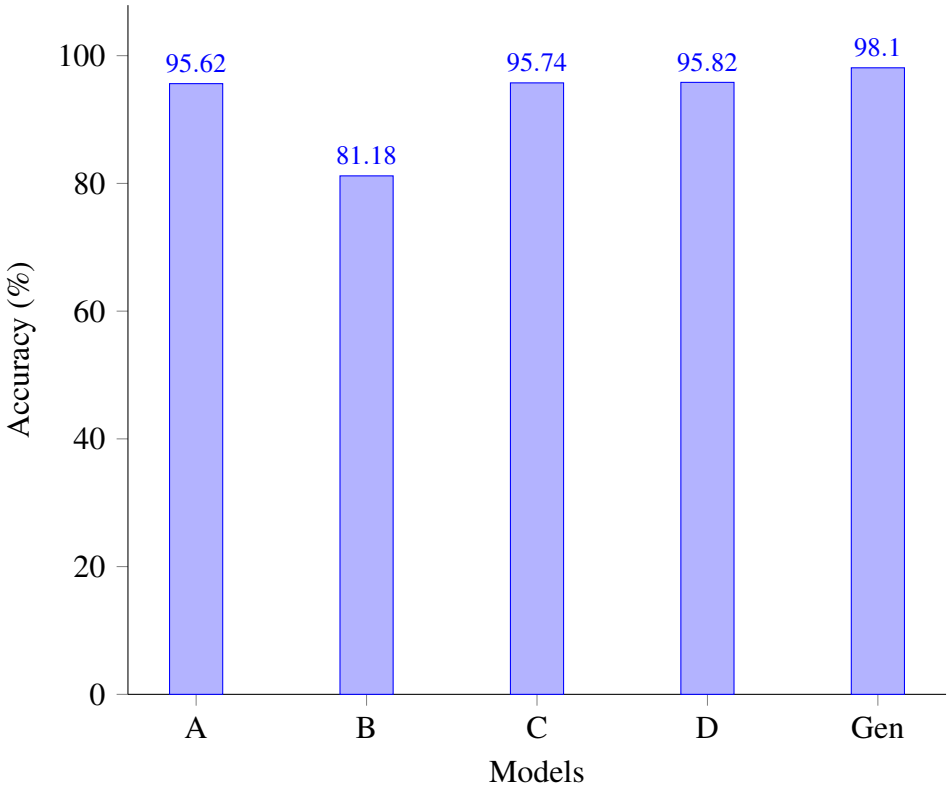

Figure 2.1: Overall classifying accuracy of all five models in the MNIST dataset. A- Model **A** , B- Model **B** , C- Model **C** , D-Model **D** , Gen- Generator

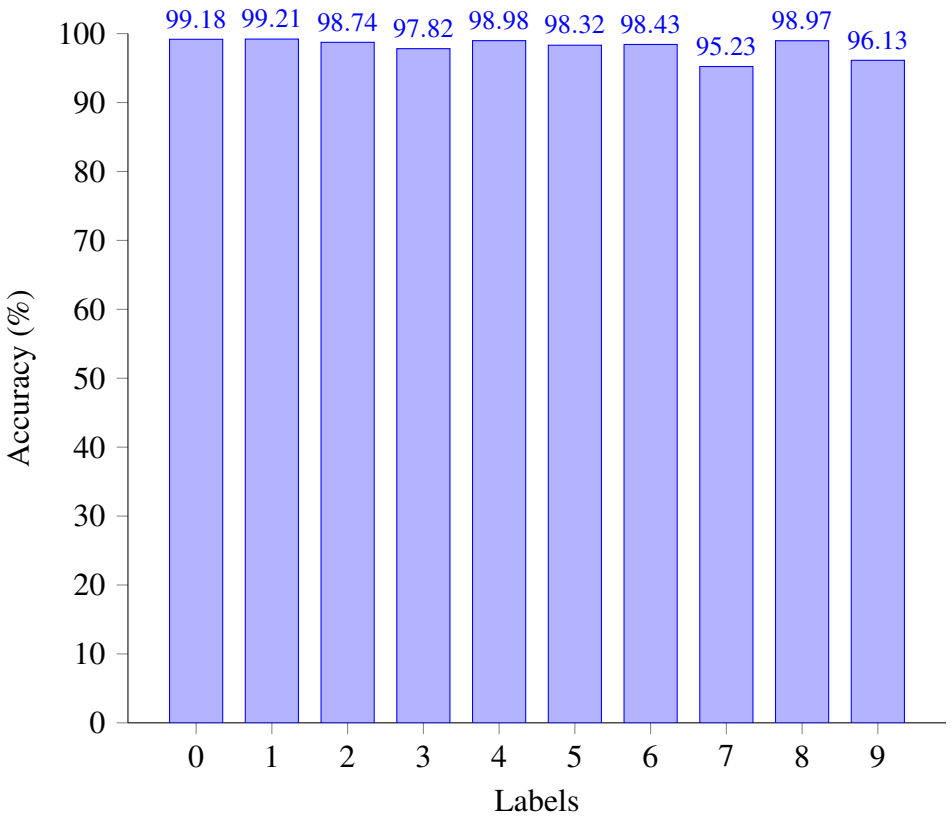

Figure 2.2: **Generator's** accuracy, per label, when classifying the MNIST dataset.

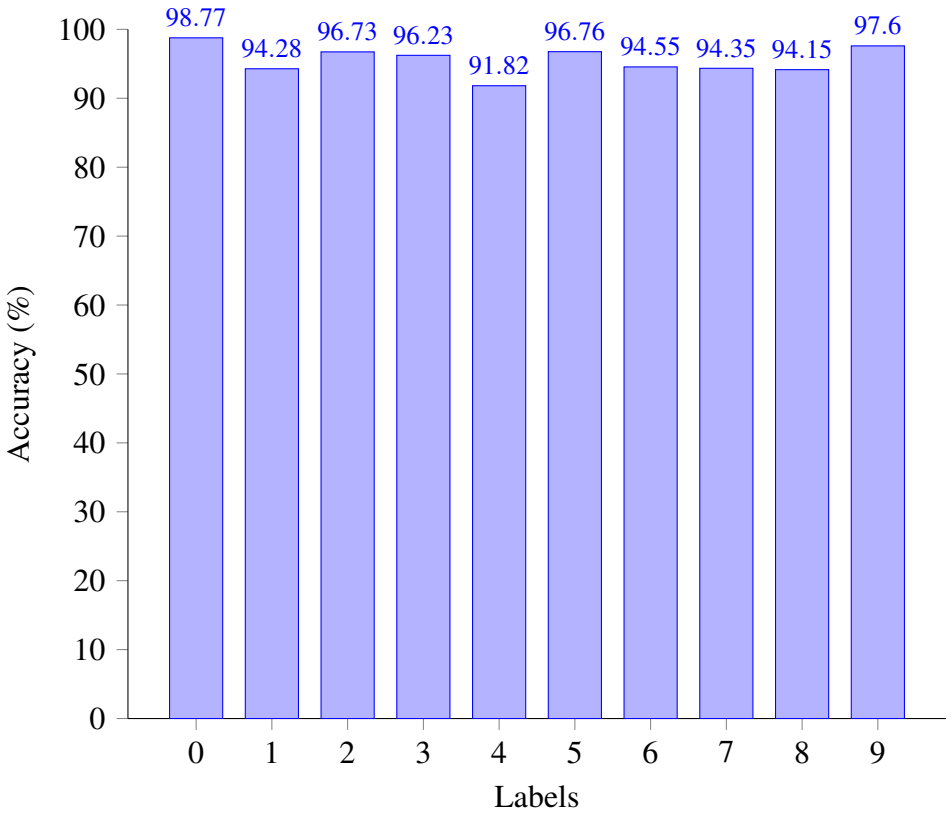

Figure 2.3: **Model A** 's accuracy, per label, when classifying the MNIST dataset.

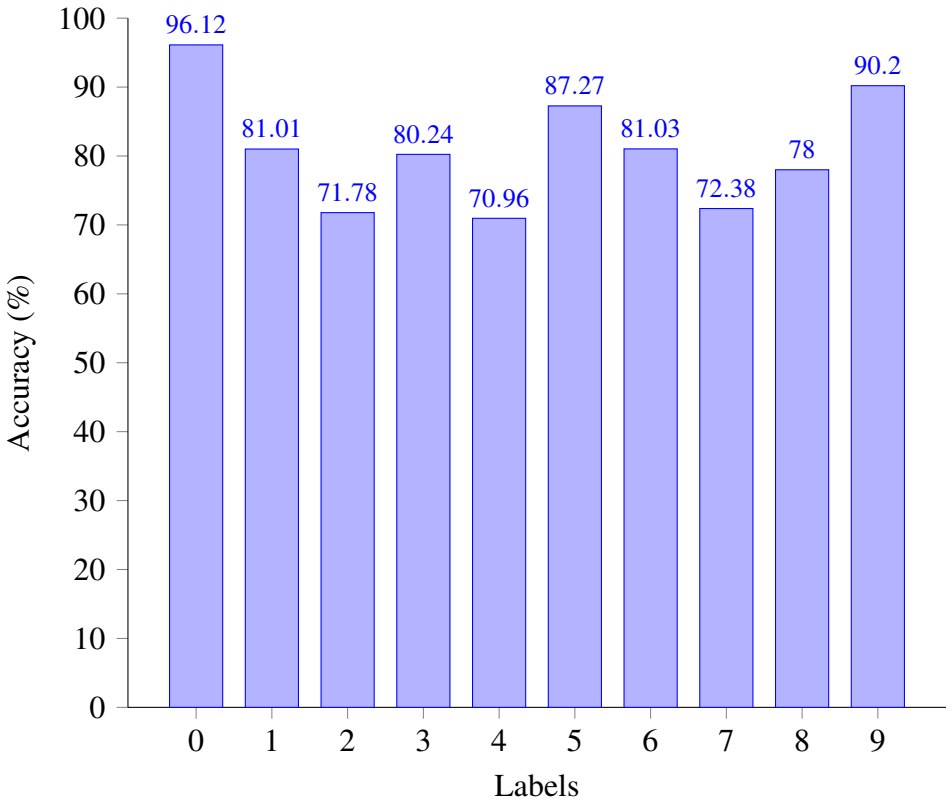

Figure 2.4: **Model B** 's accuracy, per label, when classifying the MNIST dataset.

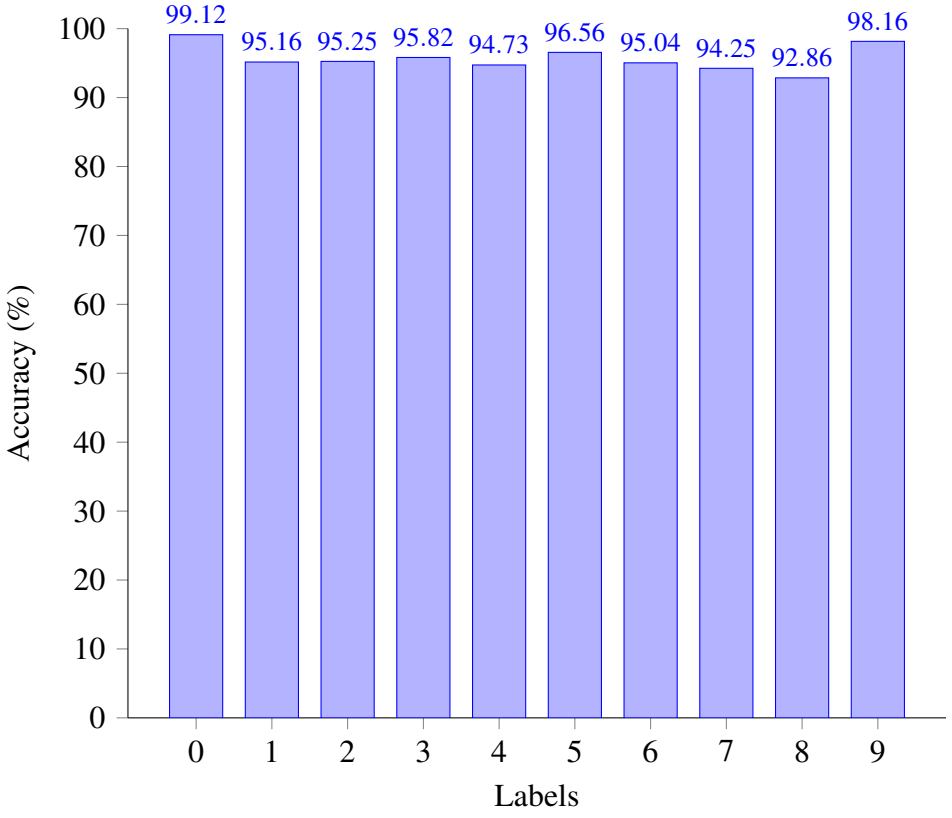

Figure 2.5: **Model C** 's accuracy, per label, when classifying the MNIST dataset.

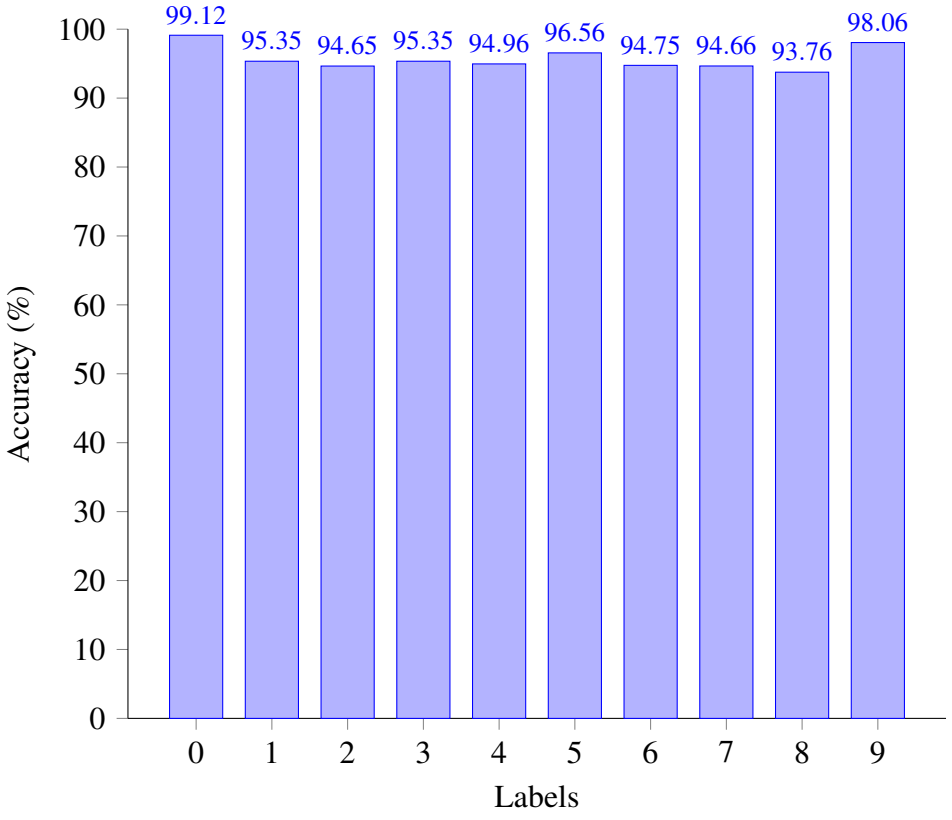

Figure 2.6: **Model D** 's accuracy, per label, when classifying the MNIST dataset.

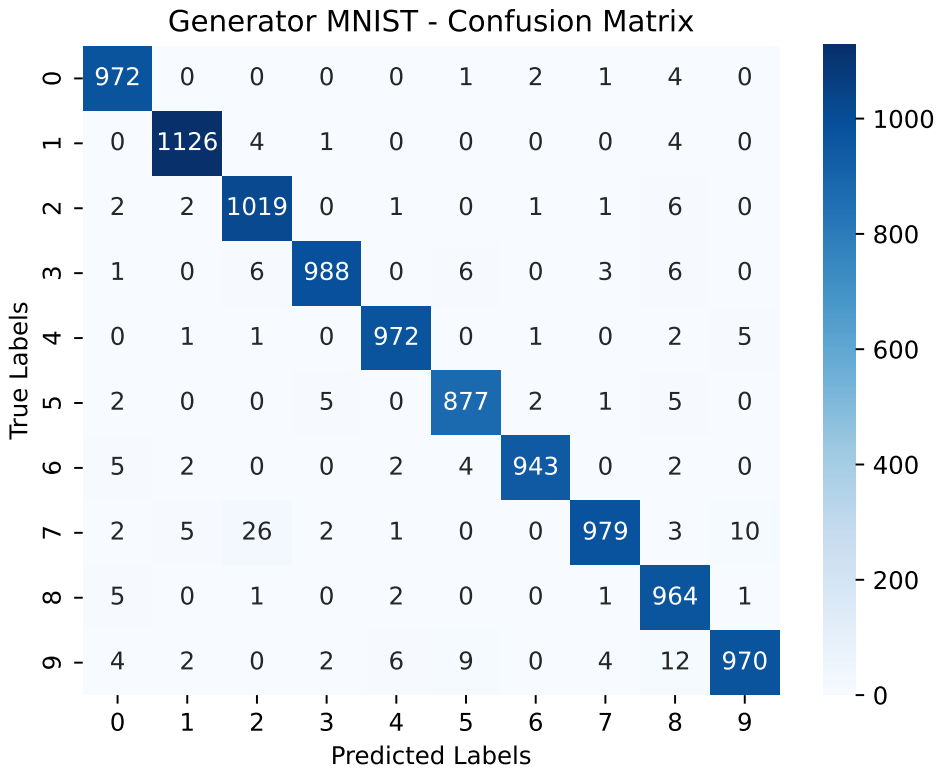

Figure 2.7: Confusion matrix summarizing **generator's** performance in the MNIST dataset classification task.

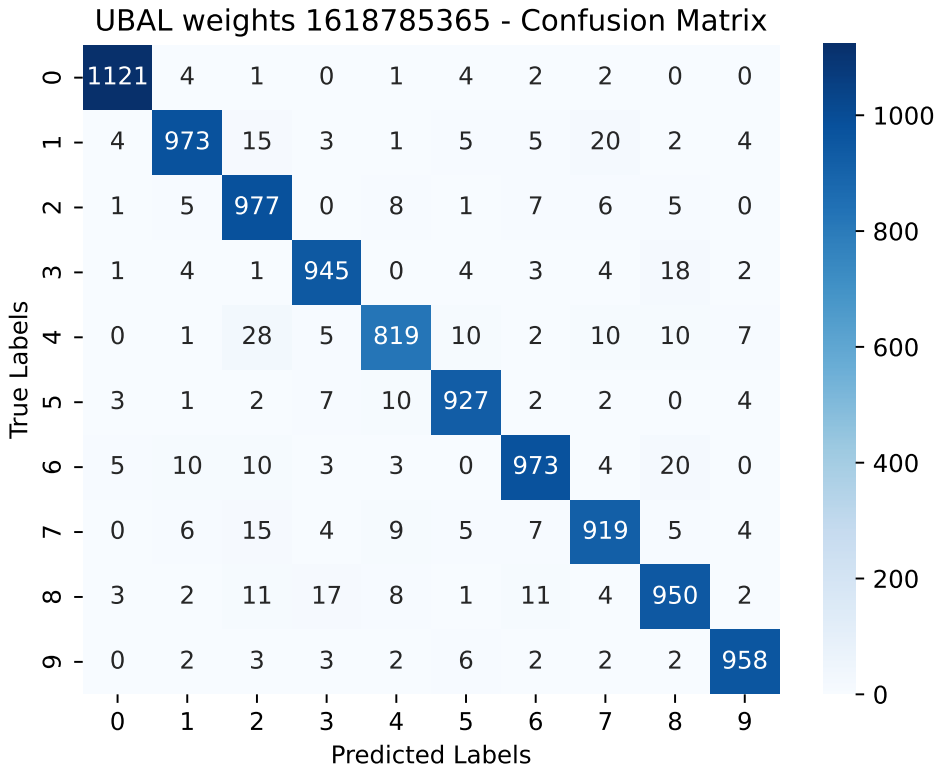

Figure 2.8: Confusion matrix summarizing **Model A** 's performance in the MNIST dataset classification task.

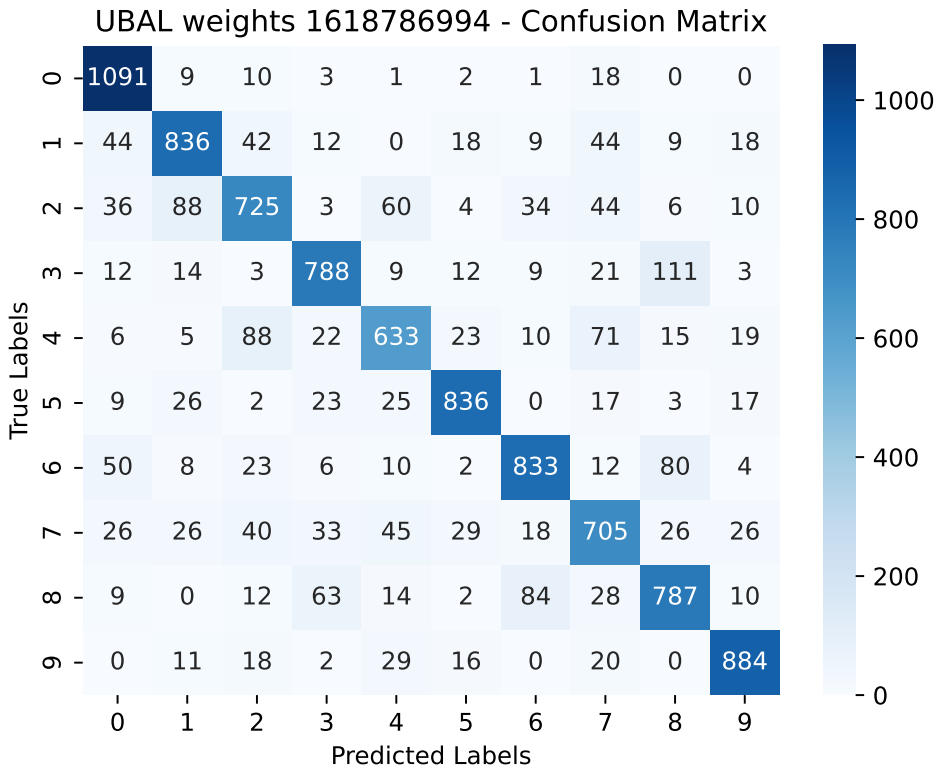

Figure 2.9: Confusion matrix summarizing **Model B** 's performance in the MNIST dataset classification task.

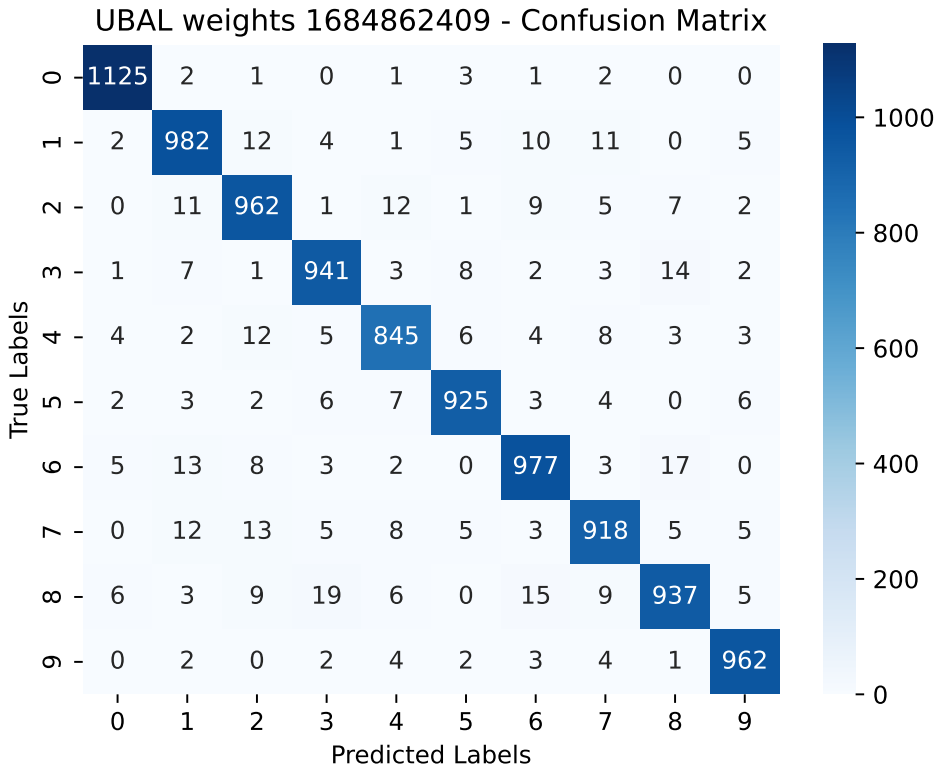

Figure 2.10: Confusion matrix summarizing **Model C** 's performance in the MNIST dataset classification task.

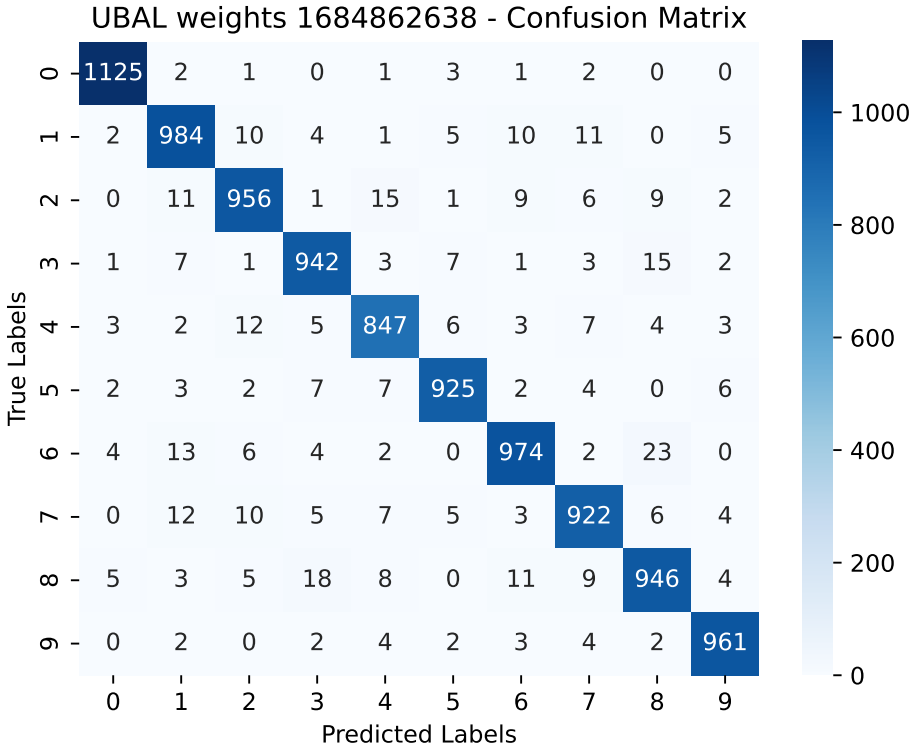

Figure 2.11: Confusion matrix summarizing **Model D** 's performance in the MNIST dataset classification task.

Based on the graph 2.1, it is evident that the generator had the highest accuracy overall. Models A, C, and D yielded similar results, with an accuracy ranging between 95-96%, while Model B had an accuracy of 81.8%. Regarding misclassifications, the generator frequently misidentified numbers 7, 9, and 3. On the other hand, UBAL models commonly misclassified numbers 4, 7, and 8. The confusion matrices were similar, except for Model B 2.9, which incorrectly identified numbers 3 as 8, 4 as 2, and 8 as 6.

2.2 UBAL's robustness against adversarial attacks

We created adversarial examples 1.9 using the pre-trained model by Pytorch [35] and attacked it using the FGSM attack 1.9. Before generating these examples, we established epsilon values, which determine the amount of noise added to the original MNIST image- the size of the perturbance of the given image. We used various ε values ranging from 0.0 to 0.5 in increments of 0.05. For each ε, we generated 3000 images, resulting in a total of 33,000 images. For assessing the networks' performance, we used accuracy when classifying the adversarial images.

Simplified attacking algorithm using the generator:

1. Initiate ε values and number of images per epsilon
2. Iterate over εs
    (a) Iterate over the MNIST data
        i. Forward pass the image data through the model
        ii. Call FGSM Attack - create the perturbed image using the epsilon
        iii. Classify the perturbed image
        iv. Save the adversarial MNIST image, label, generator's output
3. Save the data to a pickle file

Please note that the adversarial images that were generated also included images that the generator had previously classified successfully before the attack.

Once we had generated the adversarial images, we utilized them as input for the UBAL. The following pages present graphs showing the mean of how accurately UBAL classifies those adversarial images depending on the perturbation size. Each graph 2.14, 2.15, 2.16, 2.17 corresponds to a different UBAL model (refer to table 2.1). We utilized the trained generator's model for all four of UBAL's trained models, resulting in consistent accuracy across all of them. Graph 2.13 displays the overall accuracy of all four models and the generator. Afterward, we analyzed the accurate categorization of UBAL based on the label. We achieved this by generating a confusion matrix for each model and comparing it with the generator's confusion matrix. Matrices have been generated for all adversarial images, containing all 11 types of perturbations in the resulting matrices.

For a better understanding of an adversarial MNIST image, please refer to Fig. 2.12, which contains 5 original images and 30 perturbed MNIST images. The images are grouped by ε, and above each image is a label and how UBAL classified it.

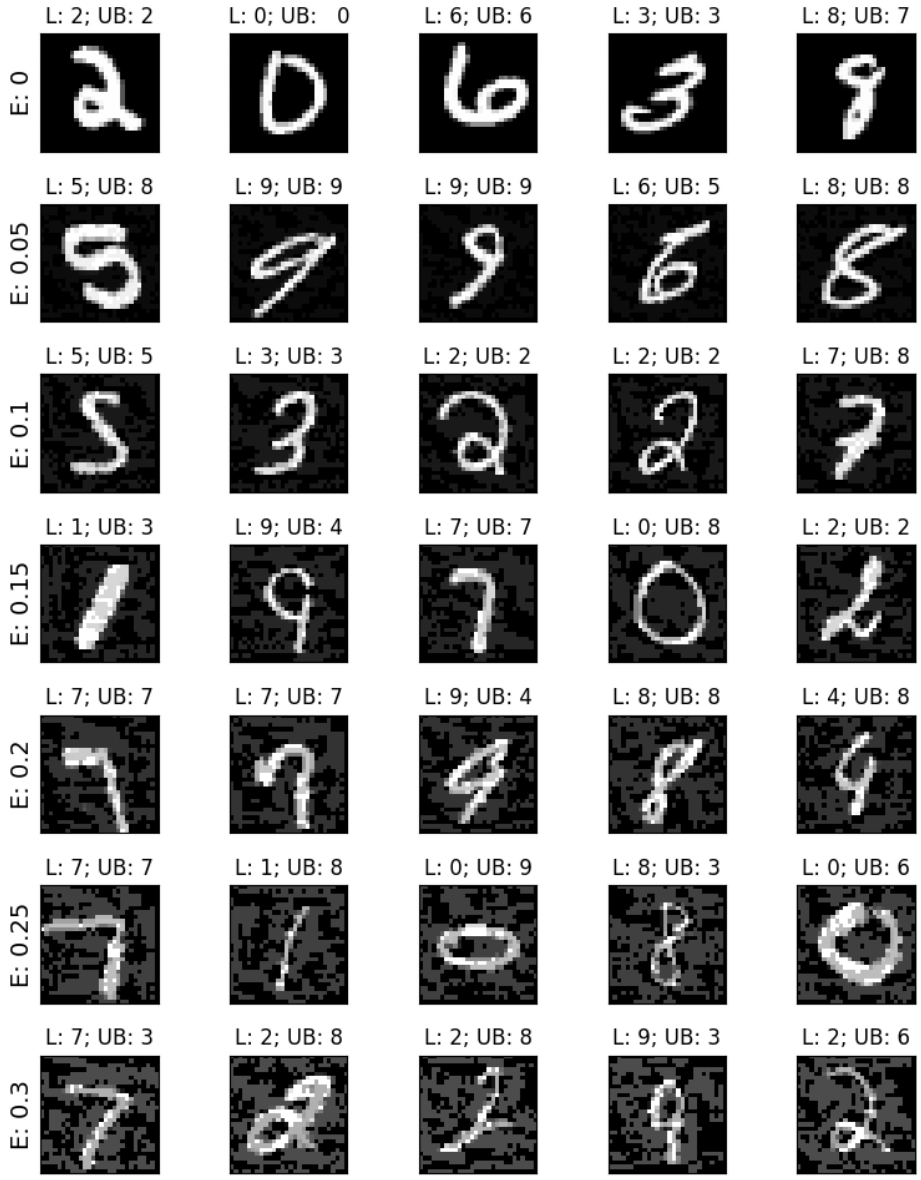

Figure 2.12: Illustration of 30 adversarial MNIST images and 5 original images (E: 0). Each row represents the degree of perturbation, i.e. ε (E). Above each image is a label (L) and how UBAL classified the image (UB).

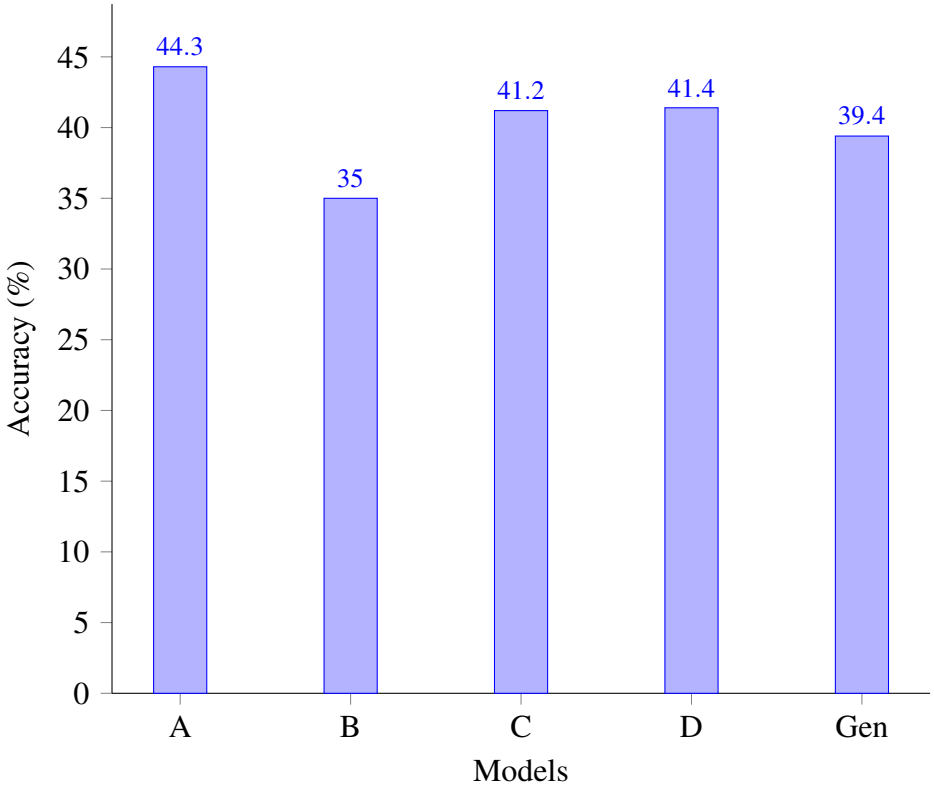

Figure 2.13: Overall accuracy of the models (all epsilons combined) when classifying the adversarial images. A- Model **A** , B-Model **B** , C- Model **C** , D- Model **D** , Gen- Generator

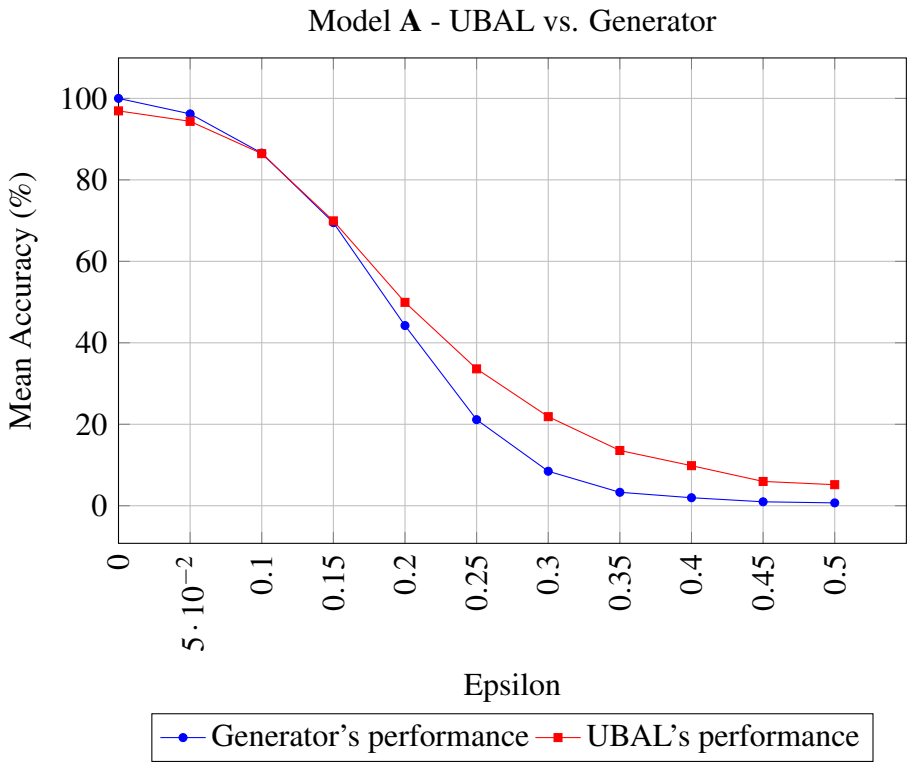

Figure 2.14: The graph displays UBAL's performance, as a function of epsilon, when trained with 1500 neurons on the hidden layer. The learning rate was 0.05, β_3=0.0, and 50 epochs.

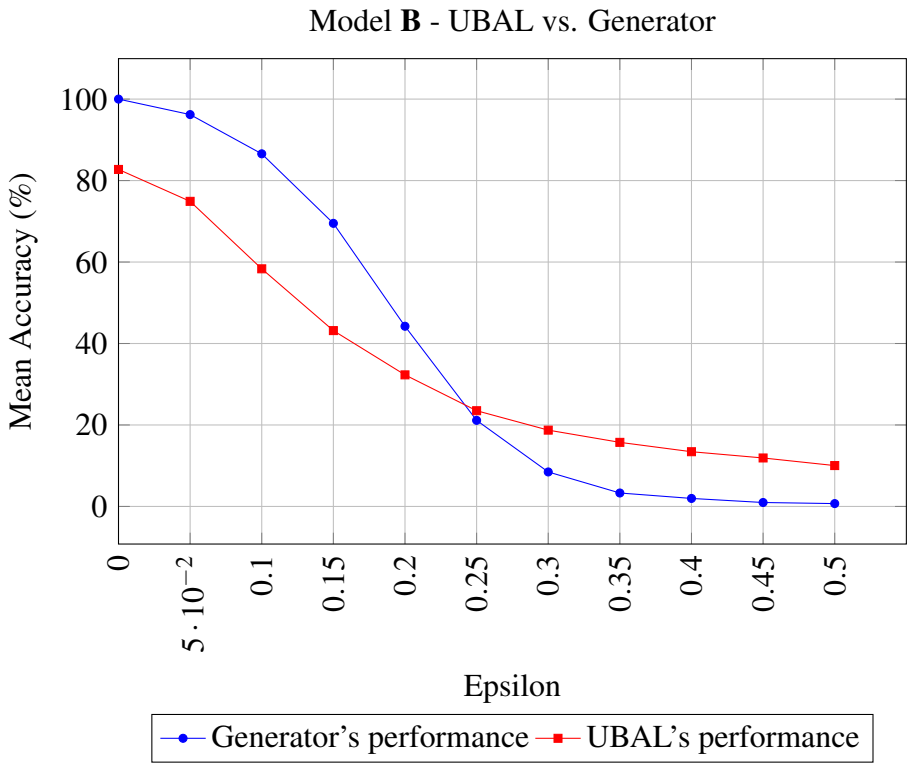

Figure 2.15: The graph displays UBAL's performance, as a function of epsilon, when trained with 1500 neurons on the hidden layer, a learning rate of 0.05, β_3=0.9, and 60 epochs.

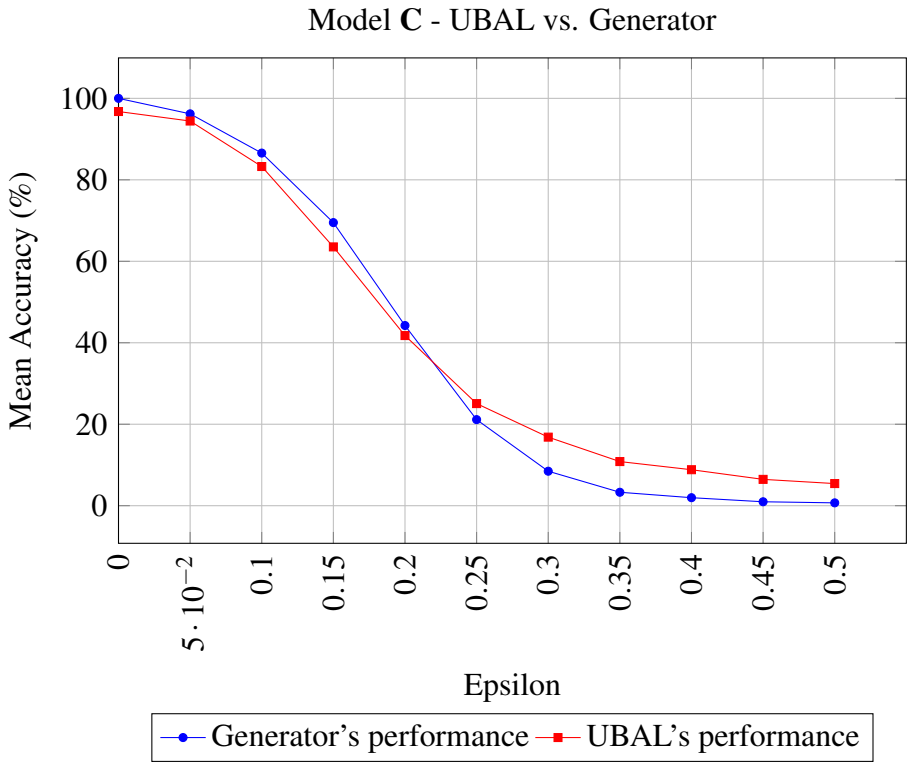

Figure 2.16: The graph displays UBAL's performance, as a function of epsilon, when trained with 1800 neurons on the hidden layer, a learning rate of 0.01, β_3=0.9, and 120 epochs.

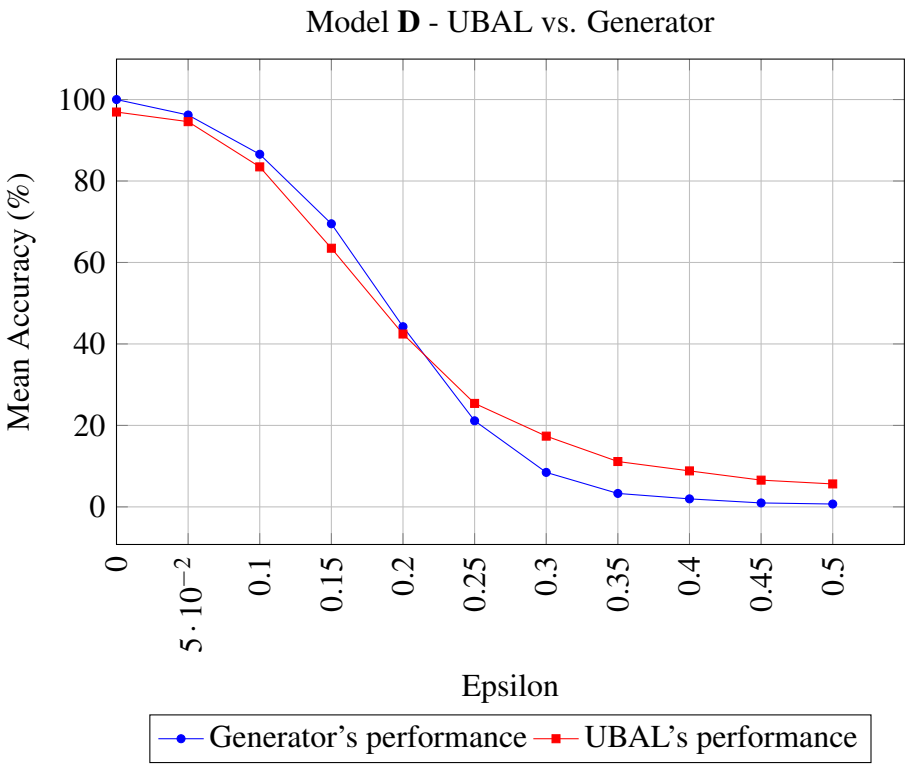

Figure 2.17: The graph displays UBAL's performance, as a function of epsilon, when trained with 1800 neurons on the hidden layer, a learning rate of 0.01, β_3=0.9, and 120 epochs.

Graph 2.13 shows that only Model **B** had less success classifying the images than the generator. However, UBAL performs better in more perturbed images; specifically, when ε >= 0.25, UBAL has higher accuracy than the generator. In addition, Model **A** performs better even from ε >=0.15, as seen in graph 2.14.

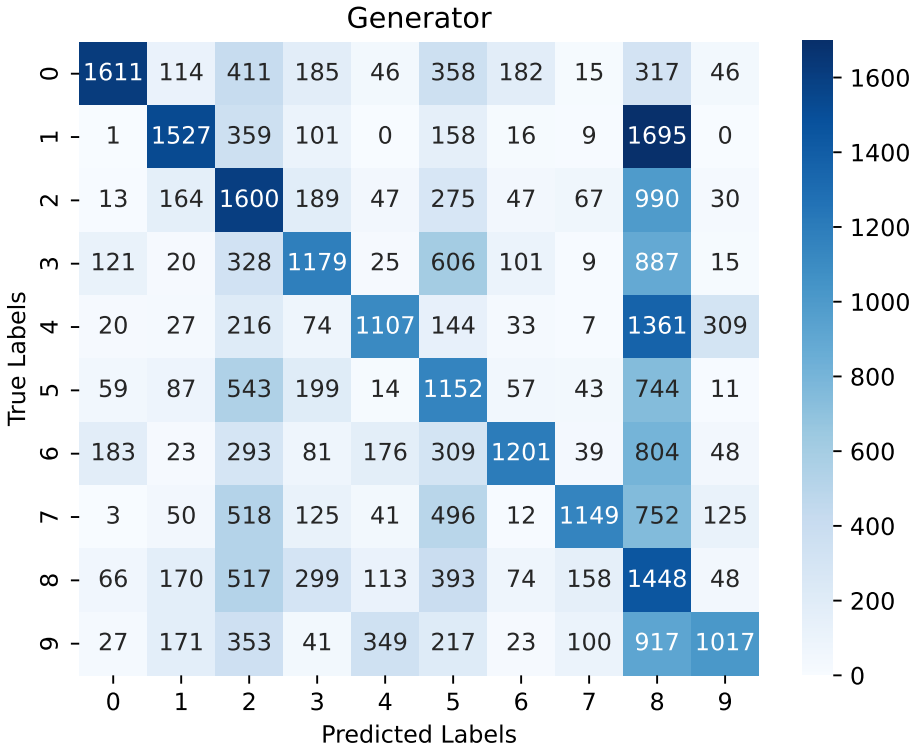

Figure 2.18: Confusion matrix summarizing the **generator's** performance in the adversarial images classification task.

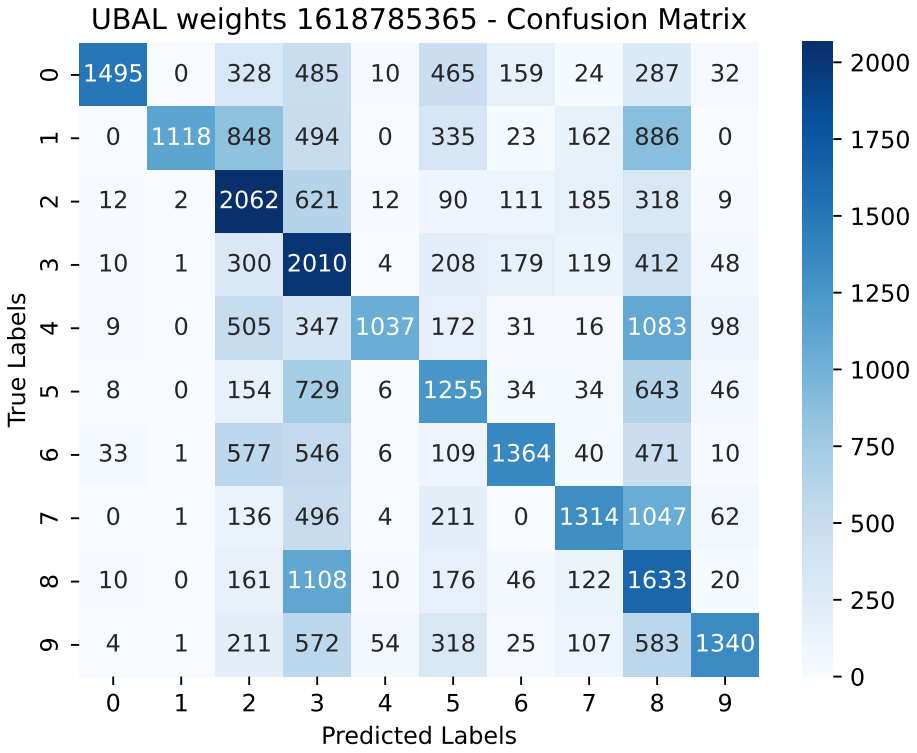

Figure 2.19: Confusion matrix summarizing **Model A** 's performance in the adversarial images classification task.

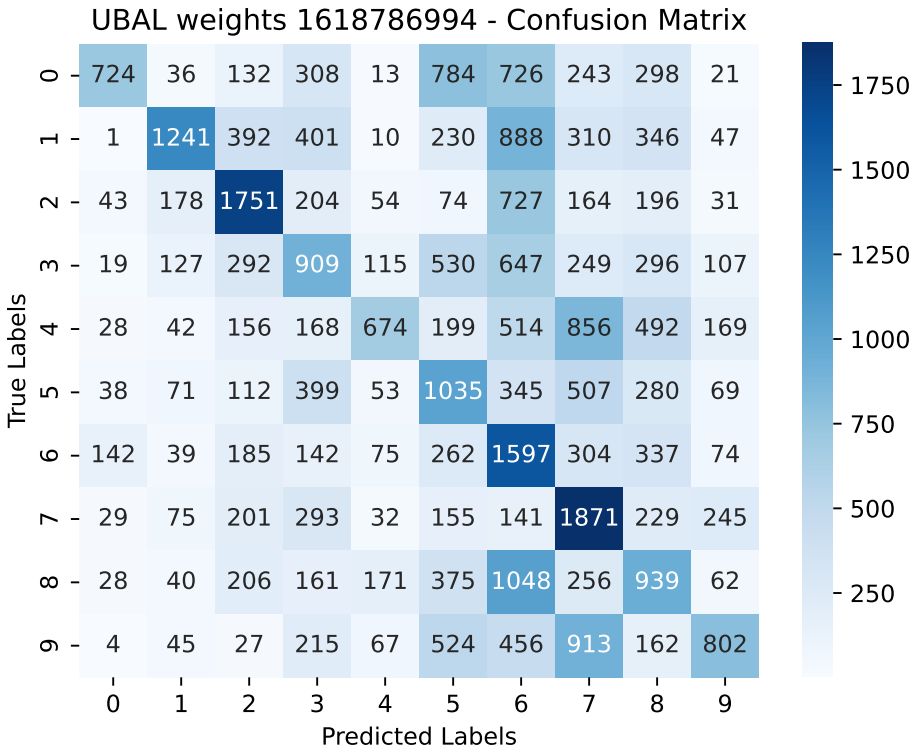

Figure 2.20: Confusion matrix summarizing **Model B** 's performance in the adversarial images classification task.

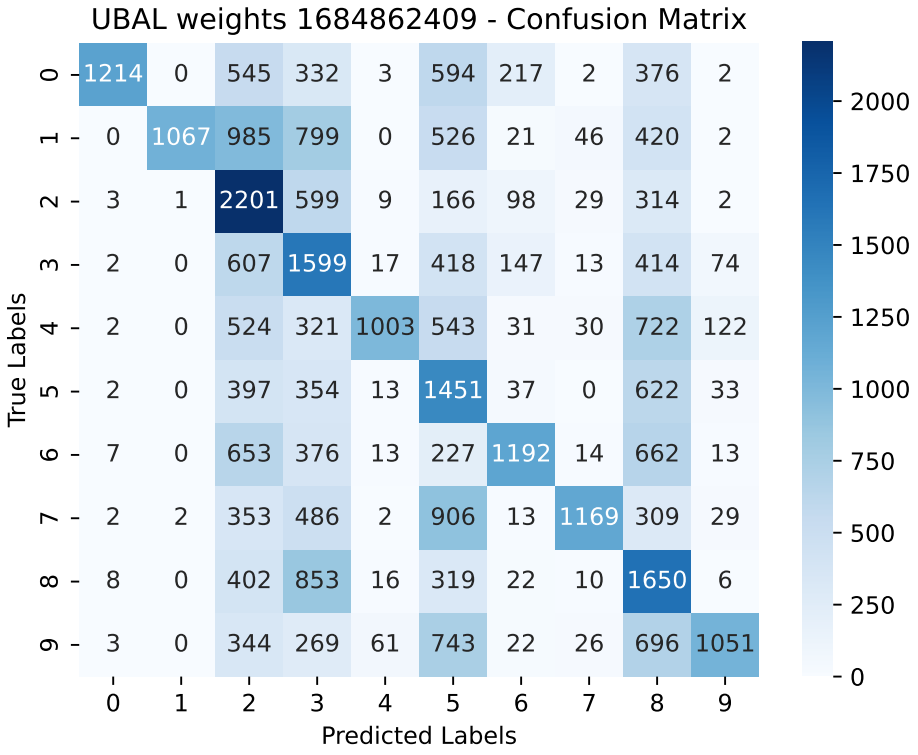

Figure 2.21: Confusion matrix summarizing **Model C** 's performance in the adversarial images classification task.

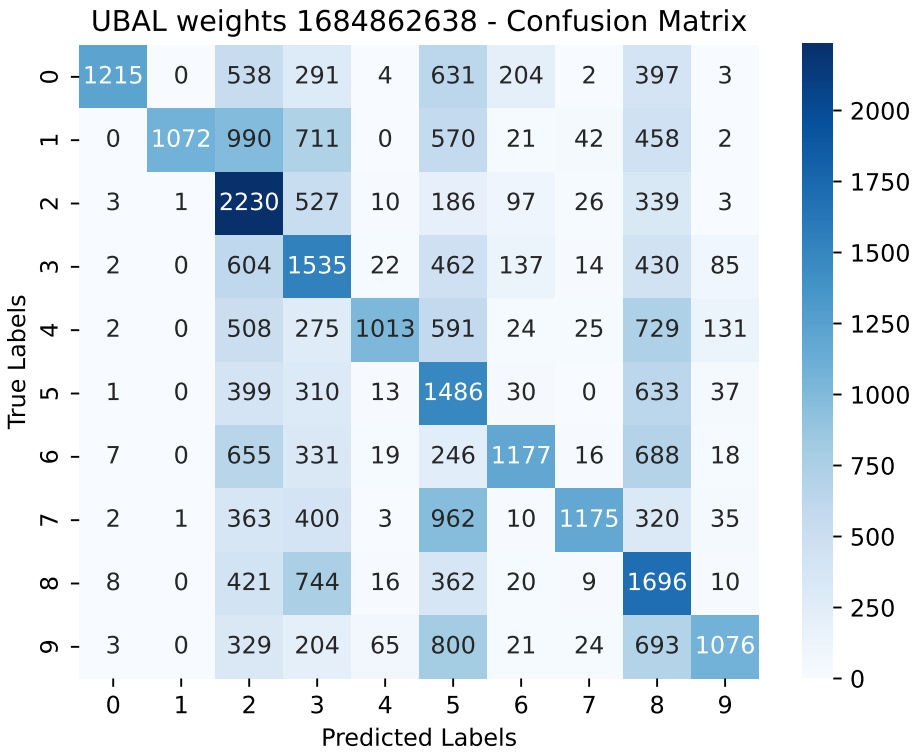

Figure 2.22: Confusion matrix summarizing **Model D** 's performance in the adversarial images classification task.

After analyzing the confusion matrix of the generator, it became apparent that the number 8 was the most frequently misclassified digit. Specifically, when the labels were 4, 1, or 3, the generator often misclassified them as 8. Regarding the UBAL models, three of the four models (A, C, D) mistakenly predicted that the label was 3 when it was 8. In addition, Model A frequently misidentified the number 8, whereas the true labels were 7 or 4. Model B tended to incorrectly predict the number 6 when the true label was 8 and the number 7 when the true label was 9. Lastly, Models C and D both tended to mistakenly predict the number 2 when the true label was 1.

2.3 Fréchet Inception Distance of UBAL's backprojections

As already mentioned in previous chapter (refer to section 1.5.4), UBAL can generate images with specific hyperparameters settings. These images are referred to as backprojections or prototypes of the input data. We wanted to explore the possibility of using UBAL as an adversarial image generator. To explore this, we again used the well-known MNIST dataset and the four UBAL models. For the FID calculation, we used a Pytorch library called *Pytorch-fid* [37]. In our experiment, we utilized two groups of images: the MNIST dataset and a modified version with added small Gaussian noise (mean = 0, variance=0.03). The Gaussian noise is illustrated in figure 2.23. Each set consisted of 1000 images.

A simplified version of calculating the FID for one model:

1. Iterate over the MNIST data
    (a) Create a .png image of the original MNIST image
    (b) Create a noisy image by adding the small Gaussian noise
    (c) Forward pass the original MNIST image through the model
        i. Save the UBAL's backprojections
    (d) Forwards pass the noisy image through the model
        i. Save the UBAL's backprojections
    (e) Run *Pytorch-fid* script for the original images
    (f) Run *Pytorch-fid* script for the noisy images

Pytorch-fid calculated the FID by comparing the original images with the backprojections. In the following pages, we will present the results of each UBAL model.

Figure 2.23: Number 3 from the MNIST dataset modified with the Gaussian noise

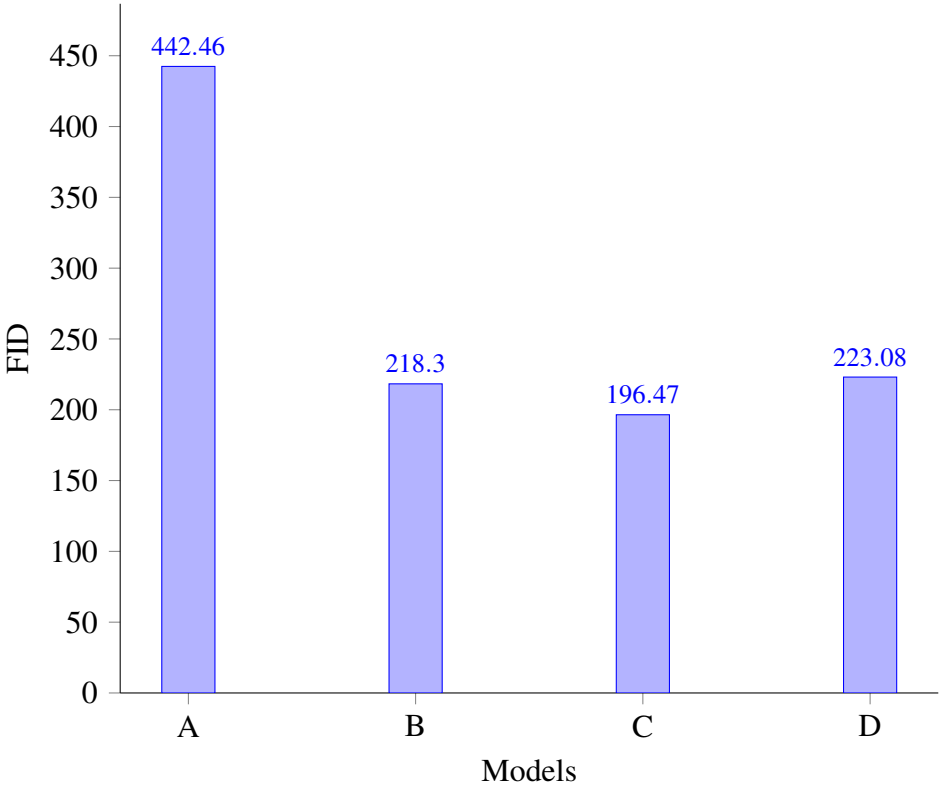

Figure 2.24: FID of all four models in the MNIST dataset. A- Model **A** , B- Model **B** , C-Model **C** , D-Model **D**

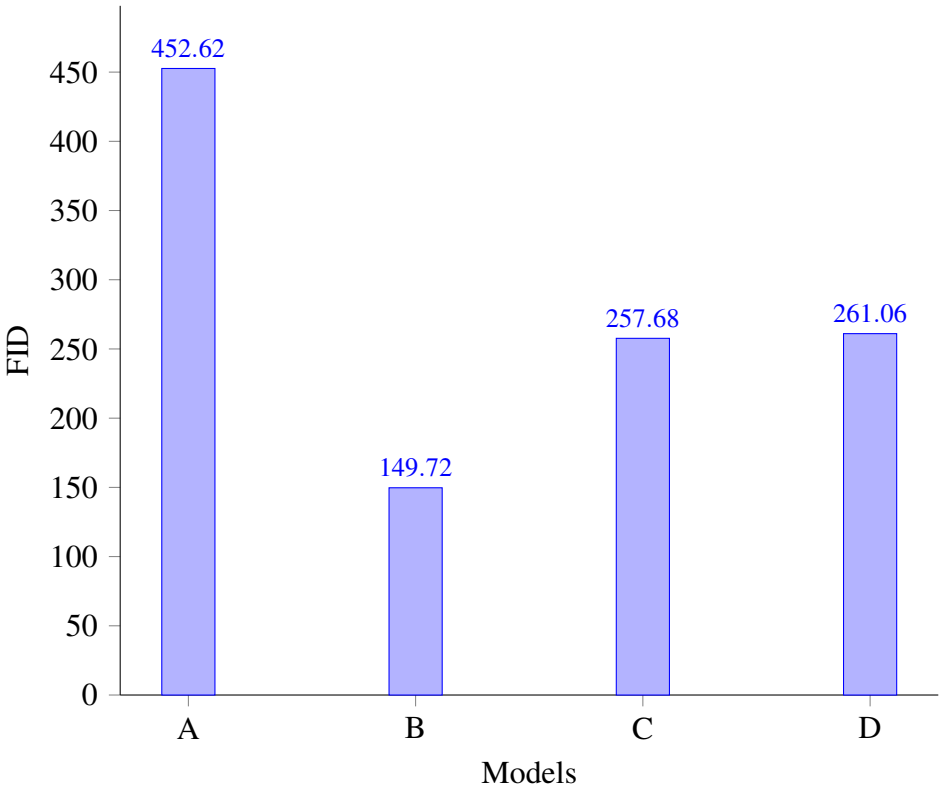

Figure 2.25: FID of all four models in the **noisy** MNIST dataset. A- Model **A** , B- Model **B** , C- Model **C** , D-Model **D**

Generally, lower FID values indicate better quality and similarity. However, what constitutes a "good" quality may vary depending on the image dataset, complexity, and desired fidelity level. Establishing a baseline or reference value specific to the dataset and task is important. In most cases, a lower FID value indicates better quality for generated images.

According to the article by Heusel [31], here are some guidelines for interpreting FID values:

1. FID < 10: The quality of the generated images is excellent, as they closely resemble real images. Additionally, there is a high level of similarity between the distributions.

2. 10 <= FID < 25: The quality of the generated images is good. They accurately capture some characteristics of the real images and maintain a reasonable level of fidelity.

3. 25 <= FID < 50: The quality is moderate, meaning that the images produced are similar to real ones but may not include small details or could contain visible flaws.

4. FID => 50: The quality of the generated images is poor as they differ significantly from real images and have a low similarity in their distributions.

Based on the bar charts, it is evident that none of the UBAL models scored low enough on FID for the images to be considered moderate. However, Model B had the lowest FID score, indicating that it produced the best results.

Discussion

In this study, we analyzed a new Artificial Neural Network model called UBAL. Our focus was on its unique features, robustness against adversarial images, and capability to generate images. We evaluated UBAL's performance by employing the MNIST dataset and utilizing the Fast Gradient Sign Method (FGSM) attack to generate adversarial images. We compared it with the model that generated the adversarial images used in this study. We also assessed UBAL's ability to generate images using the Fréchet Inception Distance (FID) metric. The following section provides a detailed analysis and interpretation of our findings.

2.4 Interpretation of the MNIST dataset classification

After analyzing the bar charts (fig. 2.3, 2.4, 2.5, 2.6), we can state that UBAL misclassifies the same numbers, which would be problematic for us people. For instance, the number 7 - is easy for us to confuse with the number 1 or the number 8, which is often confused with the number 3. We searched for a study on participants classifying images from the MNIST dataset. We found that there is still a need for such a study. It seems like there is a general consensus that such a study would be beneficial [38].

2.5 Interpretation of UBAL's robustness

The graphs (figures 2.14, 2.15, 2.16, 2.17) depicting UBAL's accuracies in each epsilon show that it is robust after a certain epsilon. When comparing accuracies per class, UBAL is more robust than the generator. Even if it makes classification errors, they are not as significant as those made by the generator when, for example, classifying the number 8. UBAL does not have a preference for any number, as it distributes the wrong answers evenly. On the other hand, the generator is more affected by added noise. UBAL is more robust than the generator, which perturbed the original images. The generator has an obvious bias, as shown in figure 2.18, unlike UBAL, which makes mistakes evenly. We can therefore assume that it is because we included the very noisy pictures since the figure represents an average from all the epsilons.

2.6 Interpretation of the FID of UBAL's backprojections

To meet the standard set by the FID metric, the UBAL scores for the images needed to be significantly lower in order to be classified as moderate.

2.7 Limitations of the study

In future research on the UBAL model, it is essential to address some of the limitations found in the current study. One issue is that the adversarial image generation relied on a single neural network and its weights. Additionally, the testing MNIST dataset used only had 10,000 numbers, and the distribution of numbers in the adversarial images set was not equal.

2.8 Future research

To deepen our understanding of the UBAL model, we recommend using adversarial examples as input for the model. Additionally, a proper MNIST study involving human classification of the dataset is necessary, as mentioned in section 2.4. Furthermore, some of UBAL's backprojections (refer to figure 1.7) are recognizable by humans. Thus, a study investigating people's ability to classify the images generated by UBAL would be beneficial.

Conclusion

We thoroughly analyzed the UBAL Artificial Neural Network model, examining its distinct characteristics, robustness against adversarial images, and generating images. To evaluate UBAL's performance, we utilized the MNIST dataset and implemented the Fast Gradient Sign Method (FGSM) attack, comparing it to a reference model for generating adversarial images. Moreover, we evaluated UBAL's image generation abilities using the Fréchet Inception Distance (FID) metric.

After analyzing the MNIST dataset classification, we noticed that UBAL made some misclassifications similar to those made by humans. For example, UBAL confused the number 7 with 1 or 8 with 3. However, it is important to mention that more research involving human classification of the MNIST dataset is required to understand these findings better.

Regarding UBAL's robustness against adversarial images, our analysis demonstrated that UBAL outperformed the generator model in terms of accuracy. While UBAL did make some classification errors, they were not as significant as those made by the generator. Unlike the generator, UBAL's wrong answers were evenly distributed across different classes, indicating a fairer and less biased approach. This indicates that UBAL is less affected by added noise and perturbations, rendering it more robust.

Furthermore, we analyzed UBAL's image generation capabilities using the FID metric. To meet the standard set by the FID metric, UBAL's scores for the generated images needed to be significantly lower. Our analysis suggests that UBAL has the potential to create images that resemble the target class, but there is still room for improvement.

Based on the results of this study, we suggest some areas for further research. One option is to use adversarial examples as input for the UBAL model to examine its performance and resilience. Conducting a study that focuses on the human classification of the MNIST dataset is also essential. This study can provide valuable insights into the similarities and differences between human and UBAL classifications. Furthermore, the recognizability of UBAL's backprojections by humans offers an opportunity to explore the human ability to classify images created by UBAL.

In conclusion, this study has shed light on the unique features, robustness, and image generation capabilities of the UBAL Artificial Neural Network model. The findings demonstrate UBAL's alignment with human perceptual tendencies, robustness against adversarial attacks, and potential for generating images that resemble the target class. Nevertheless,

more investigation is needed to overcome this study's limitations and better understand the UBAL model.

Bibliography

[1] R. C. O'Reilly, Y. Munakata, M. J. Frank, T. E. Hazy, and Contributors, *Computational Cognitive Neuroscience*. Online Book, 4th Edition, URL: `https://CompCogNeuro.org`, 2012.

[2] I. Farkas and K. Rebrová, "Bidirectional activation-based neural network learning algorithm," 09 2013.

[3] K. Malinovská, L. Malinovský, P. Krsek, S. Kraus, and I. Farkaš, "Ubal: A universal bidirectional activation-based learning rule for neural networks," in *Proceedings of the 2019 2nd International Conference on Computational Intelligence and Intelligent Systems*, CIIS '19, (New York, NY, USA), p. 57–62, Association for Computing Machinery, 2020.

[4] K. Malinovska and I. Farkaš, "Generative properties of universal bidirectional activation-based learning," in *Artificial Neural Networks and Machine Learning – ICANN 2021* (I. Farkaš, P. Masulli, S. Otte, and S. Wermter, eds.), (Cham), pp. 80–83, Springer International Publishing, 2021.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[6] P.-T. Yap, X. Jiang, and A. Kot, "Two-dimensional polar harmonic transforms for invariant image representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 1259 – 1270, 08 2010.

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[8] D. Hebb, *The Organization of Behavior*. Psychology Press, Apr. 2002.

[9] B. Goodfellow, Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618," *Genetic Programming and Evolvable Machines*, vol. 19, 10 2016.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] D. Rumelhart, G. Hinton, and R. Williams, *Learning internal representations by error propagation*, pp. 318–362. Cambridge, MA: The MIT Press, 1986.

[12] F. Crick, "The recent excitement about neural networks.," *Nature*, vol. 337, no. 6203, pp. 129–132, 1989.

[13] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive science*, vol. 11, no. 1, pp. 23–63, 1987.

[14] P. Mazzoni, R. A. Andersen, and M. I. Jordan, "A more biologically plausible learning rule for neural networks.," *Proceedings of the National Academy of Sciences*, vol. 88, no. 10, pp. 4433–4437, 1991.

[15] R. O'Reilly, "Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm," *Neural Computation*, vol. 8, no. 5, pp. 895–938, 1996.

[16] D. E. Rumelhart and J. L. McClelland, *Learning Internal Representations by Error Propagation*, pp. 318–362. 1987.

[17] P. Csiba and I. Farkaš, "Computational analysis of the bidirectional activation-based learning in autoencoder task," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, pp. 1–6, IEEE, 2015.

[18] G. E. Hinton and J. L. McClelland, "Learning representations by recirculation," in *Neural information processing systems*, pp. 358–366, 1988.

[19] X. Xie and H. S. Seung, "Equivalence of backpropagation and contrastive hebbian learning in a layered network," *Neural computation*, vol. 15, no. 2, pp. 441–454, 2003.

[20] B. Scellier and Y. Bengio, "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation," *Frontiers in computational neuroscience*, vol. 11, p. 24, 2017.

[21] R. C. O'Reilly, D. Wyatte, and J. Rohrlich, "Learning through time in the thalamocortical loops," *arXiv preprint arXiv:1407.3432*, 2014.

[22] R. C. O'Reilly, Y. Munakata, M. Frank, T. Hazy, *et al.*, *Computational cognitive neuroscience*. PediaPress, 2012.

[23] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, p. 13276, 2016.

[24] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," 2017.

[25] O. Press, A. Bar, B. Bogin, J. Berant, and L. Wolf, "Language generation with recurrent generative adversarial networks without pre-training," 2017.

[26] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[27] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[28] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[29] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[30] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," 2016.

[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[32] M. Fréchet, "Sur la distance de deux lois de probabilité," *Annales de l'ISUP*, vol. VI, no. 3, pp. 183–198, 1957.

[33] L. N. Vaserstein, "Markov processes over denumerable products of spaces, describing large systems of automata," *Problemy Peredachi Informatsii*, vol. 5, no. 3, pp. 64–72, 1969.

[34] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[35] N. Inkawhich, "Adversarial Example Generation 2014; PyTorch Tutorials 2.0.1+cu117 documentation — pytorch.org." `https://pytorch.org/tutorials/beginner/fgsm_tutorial.html`. [Accessed 22-May-2023].

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[37] M. Seitzer, "pytorch-fid: FID Score for PyTorch." `https://github.com/mseitzer/pytorch-fid`, August 2020. Version 0.3.0.

[38] "What is human accuracy on the MNIST test set? Are there any quotable sources? — quora.com." `https://www.quora.com/What-is-human-accuracy-on-the-MNIST-test-set-Are-there-any-quotable-sources`. [Accessed 02-Jun-2023].

Appendix

The code is attached to the electronic version of this thesis.