TRANS-
LATION
OF COM-
PUTATION

CONRAD WEISE
INTERMEDIATE I

BACHELOR INTEGRATED DESIGN

AREA OF EXPERTISE IAD
UNDER THE SUPERVISION OF
PROF. DR. LASSE SCHERFFIG

KÖLN INTERNATIONAL SCHOOL OF DESIGN
FACULTY OF CULTURAL STUDIES

CONTENTS

COM- PUTA- TION

ARTICLE 01

Computation is all around us. Since the breakthrough of modern integrated circuit technology, computation expanded into every aspect of everyday life. Transistor sizes down to 7 nanometer enable computation to hide behind user interfaces of all kinds, the internet creates a vast mesh of computational networks that connect and govern over half of the world‘s population, and companies chosen to bet on computation are now the leading top of economical power (Bridle, 2018).

Computation is serving this platform of utter most power to its instructionist, the software. Software is the determining part of the computational equation. Its abstract symbol structures will guide the physical logic gates, which will inevitably result into a ripple of multiple electrical signals, deciding about the final output of the computational equation. Whether the final output is about displaying some trending tweets or whether it is consulting a judge deciding about someone‘s prison sentence; software is part of the equation and it is not only part of it, but it constitutes most of it. Through the computational platform software has the essential scale and the necessary flexible physical structure to be embedded into everyday life. Instead of just being part of it, being the layer on top or down under of the physical world, it has proven itself to be the interwoven medium used to build the systems of the information age of today. Its existence became essential to the western modern life.

While computation (from now on as software and physical computation intertwined) decided more and more about human life, showing its increasing imprint in socio-political implications and with progress in the area of Artificial Intelligence, it gained increasing attention from sociologists and media theorists reflecting on the state of software in our societies. With the breakthrough of machine learning and deep neural networks and their immediate wide usage one particular phenomenon was recently acknowledged more frequently:

Biased algorithms describe the appearance of discrimination and the reinforcement of prejudice connected to identity constructs like race, ethnicity, sexuality, gender or culture in computational systems. The concept of algorithmic bias rose in popularity because of recent incidents of heavily biased machine learning algorithms (Lipton, 2016).

As we will see throughout the following articles, the idea is that the problem of algorithmic bias is a more complex one that shouldn‘t be simplified and limited to the discourse of machine learning. Although machine learning brings its new technical concepts with it, which shouldn’t be disregarded in the analysis of bias, it is important to note that algorithmic bias is rooted in software per se, not machine learning. Algorithmic bias is a fundamental software problem. In recent conversations machine learning served as a catalyst and opaque layer which enforced the bias and distracted most research away from the actual underlying flaws of software development.

To understand the implications of bias in machine learning it is important to first look at the foundation of bias in traditional software.

OBJEC-TIVI-TY OF MATH

ARTICLE 02

The previously mentioned underlying computational platform, on which software is building upon, can be reduced to elementary mathematics. The base component of modern computer systems is the integrated circuit. It consists of numerous transistors, which form logic gates, being the main logic operator serving the defined algorithms of the software. The transistors in a logic gate essentially function as a switch, making it possible to execute the most basic logical operations, like NOT, AND, OR but also NAND, NOR, XOR, XNOR. Within the computational platform, they serve as the fundamental implementation of software. From there on the complexity of logic is as good as infinite. All complexity in modern computer systems is based upon and broken down in these basic operations to be able to execute and understandable to the computational platform (Sangosanya, Belton and Bigwood, 2005).

An algorithm defined in software is hard to generalize and define, because of the concept's wide diversity of usage. Nevertheless an algorithm always defines given input data, defines instructions on how to process that data, and produces an output based on the given instructions. While algorithms can easily exceed human comprehension in complexity, its basic structures are always the same and based upon the logical operations of the underlying logic gates. In a simplified way, looking at decision tree diagrams and the control flow of an algorithm, every new stage of the algorithm results in a yes or no answer based on specific statement, like IF and ELSE, which are again represented in objective logical operations.

The nature of algorithms is objective math; and with that goes the general assumption that computation has some kind of privileged status of objectivity. That computer systems are based on mathematical hard facts and therefore are free of human prejudice or interpretation of any kind, blinded by the mere assumption of the algorithm's neutrality.

While a single logic operator is funded in objective math, it is the constellation of them, which are inherently subjective. The constellation of them is neatly curated by the human creator and with that every statement of the algorithm constitutes a formalized representation of the creator's environment, views and prejudice.

> Computers neither consider nor generate facts. They manipulate symbolic representations that some person generated on the belief that they corresponded to facts. (Winograd and Flores, 1987, p. 136)

The following articles will serve as the foundation of the analysis and exploration on how these representations are being implemented and where the human bias manifests itself in computation.

COM-
PUTA-
TIONAL
THIN-
KING

ARTICLE 03

For this and the following chapters I would like to introduce the work of Terry Winograd and Fernando Flores Understanding Computers and Cognition: A New Foundation for Design from 1986. The book was published in a time when computers started to receive increasing attention from any kind of business, institution or even individual consumers. Computer systems were the new technology, which was promising to drive efficiency and profit forward. In the vast progress of these new developments, Winograd and Flores proposed a new design approach, looking at the fundamentals of what it means to create computational systems and how they and their implications fit into modern society. Actively criticizing the occurring software development processes at the time. Their approach was highly influenced by philosophical and phenomenological discourses, which led them to an analysis of the core of how human cognition and computation join forces in computer systems.

While current development environments are layered and layered by different code structures and codebases, and with non-transparent machine learning techniques in mind, Winograd and Flores ontological conceptual tools become an essential part of the examination of algorithmic bias; a glance at how software's concepts are actively shaping the way we think.

Winograd and Flores open up their work by introducing the term rationalistic tradition. They use the word tradition in a different context than usual. Winograd and Flores use it to pinpoint that no question can be raised from a neutral or objective standpoint. They redefine the use of the word tradition to a broader sense, more like a pre-understanding or a way of being; more of a cultural background or way of thinking.

Their first important insight about the rationalistic tradition is that it has been the mainspring of western science and technology. Especially the ones where principles can be captured in formal systems. They state that it has greatly influenced the development of linguistics and cognitive psychology and define it further with listing a series of steps, which resemble the rationalistic orientation. To the question "What do you do when faced with some problem whose situation you care about?" the solution process would be the following:

1. Characterize the situation in terms of identifiable objects with well-defined properties.

2. Find general rules that apply to situations in terms of those objects and properties.

3. Apply the rules logically to the situation of concern, drawing conclusions about what should be done. (Winograd and Flores, 1987, p. 15)

Looking at this list it becomes apparent that problem-solving procedures like the ones shown here are of fundamental nature of business operations, scientific methods, and also software development. Just like computation entered our daily lives, the rationalistic formalization process was directly embedded into it. Computation reinforced the rationalistic tradition into every aspect of modern life.

As we will see in the following articles, this list is a dangerously simplified description of the world around us. Separating the world into objects and non-objects and assigning them properties or deciding to leave out specific properties and then generalizing rules on top of them raises numerous difficulties.

While Winograd and Flores showed high concern with the wide implementations of the rationalistic tradition, looking at contemporary literature, it seems that the rationalistic tradition was just the beginning of a new way of thinking:

James Bridle, an artist and writer, published New Dark Age: Technology and the End of the Future in 2018, where he is describing his interpretation of computational thinking. Although the term is used frequently in and originated in the computer science industry, Bridle uses the term differently and puts it into another context. Rather than seeing computational thinking as the concept of being able to abstract a problem and automate and analyse the solution, Bridle connotes the term fairly negative and sceptic. He sees

> computational thinking [as] an extension [...to] solutionism: the belief that any given problem can be solved by the application of computation. (Bridle, 2018, p. 4)

and argues that

> computational thinking is predominant in the world today, driving the worst trends in our societies and interactions [...]. (Bridle, 2018, p. 4)

Bridle goes as far as Winograd and Flores and describes it as a way of thinking:

> Computation replaces conscious thought. We think more and more like the machine, or we do not think at all. (Bridle, 2018, p. 43)

Bridle's computational thinking could be described as a highly perverted version of the rationalistic tradition. Not only do we see the world as a computational system, we actively constitute it as one. Through computation, the rationalistic tradition not only drives most of problem-solving disciplines from today, but it became the only option we have.

HERME-NEUTIC CIRCLE AND THE FALLACY OF INSTRUC-TIVE INTER-ACTION

ARTICLE 04

To further expose the core of why the rationalistic tradition aka. computational thinking poses as the killer of a fair and diverse world, this article will briefly introduce two conceptual tools to explore the difficulties and fallacies of interpretation in language and cognitive representation.

Hermeneutics was first and foremost the tool of understanding the meaning of mythical and religious text. It emerged as a theory and later became applied science, nowadays mostly camouflaged as analytics, because specific text pieces would be understood differently each time they would be re-read, realizing that the meaning of text is not absolute. Although hermeneutics was also used to completely decontextualize text in the most objectivist sense, we will use hermeneutics to take a closer look at how interpretation plays a vital role in understanding.

In the discourse of how bias is defined in software, we find appeal in the work of philosopher Hans-Georg Gadamer. In his understanding of hermeneutics, interpretation plays a vital role. It is the primary action taking place within the background provided by the reader and the background provided by the writer. The writer and reader essentially join in an interaction of understanding based on their pre-understanding of the world. For him, in contrast to the objectivists, a text cannot exist without a context and cannot be understood without regarding the context of reader and writer. For Gadamer

> any individual, in understanding his or her world, is continually involved in activities of interpretation. (Winograd and Flores, 1987, p. 28)

And just like in reading a text, these interpretations are based on the pre-understanding of the individual, fundamentally grounded in prejudice and bias. He argues that our pre-understanding and the connected prejudices and biases are inevitable. It would be naive to think, that we can perceive a situation with being fully aware of our pre-understanding. It is a fundamental part of being human to be blinded by our own background (Winograd and Flores, 1987).

The important insight I want to introduce, is the concept of Gadamer‘s hermeneutic circle. With that he closes the loop encapsulating the feedback loop of pre-understandings. While I read a text, I can only understand it with my particular background in time. However this background is just another product of other interpretations I made before based on the pre-understanding I possessed at that time.

> What we understand is based on what we already know, and what we already know comes from being able to understand. (Winograd and Flores, 1987, p. 30)

To further look into the understanding of human perception, the biologist Humberto Maturana presents very interesting

findings questioning the validity of common sense understanding of biological cognition.

In the work “Anatomy and Physiology of Vision in the Frog” (Maturana, Lettvin, McCulloch and Pitts, 1960) Maturana, with three other biologist, first discovered and shed light on the misconception of direct cognitive representations. In this work they stated that the activity in the optic nerve of a frog was not a direct representation of the light pattern on the retina. As it turned out specific fibers, connected to the retina, already took over specific cognitive processes. One type of fiber, for example, responded best to flies, being triggered by a small dark spot surrounded by light (Maturana, Lettvin, McCulloch and Pitts, 1960).

With more research and revelations of that kind, Maturana went as far as calling it the fallacy of instructive interaction, meaning that direct cognitive representation of our environments are a big misconception. Supporting the statement with his concept of autopoiesis and structural coupling, describing that human interaction with the environment is always happening through the entire nervous system. Changes that are happening in the perceived environment are not represented one by one in the nervous system. The interconnection between perceiving and the saved state in the nervous system are not representational.

With that in mind Winograd and Flores go on to counter Maturana’s findings to the behaviorist approach to cognition, stating that an organism behaves on the ground on external stimuli. They decipher his argumentation as following:

> Maturana [...] argues that we cannot deal with organism and environment as two interacting independent things. We cannot identify stimuli that exist independently of the unity and talk about its history of responses to them. The unity itself specifies the space in which it exists, and in observing it we must use distinctions within that space. (Winograd and Flores, 1987, p. 48)

While the digression into the topic of hermeneutics and cognitive science doesn’t seem relatable to algorithmic bias, as we will see in the following article, the described fallacies of interpretation and cognitive representation are the foundation within software development.

REPRE- SENTA- TION

ARTICLE 05

While programming languages started out to be very close to actual machine code, being 1s and 0s, which the boolean logic gates could directly interpret, programming languages nowadays oriented themselves through concepts like object oriented- and functional programming towards written language. So called high-level languages are closer to English than to machine code. Although it seems like the implementation of high-level programming languages orients itself away from its computational origin, it is important to note that it is just another abstraction sitting on top of the highly rational structure of computation.

Nevertheless, software can and should be treated as language to fully understand its affiliation with hermeneutics. Not to forget, that in practice, writing software is an inherently social activity. Humans write code for humans. Within contemporary software development, codebases are managed by numerous programmers at the same time. They engage in a conversation of formalized views of the world, being in constant flux of their interpretations and cognitive representations. Reading code becomes an essential part of software development and with that also the interpretation and understanding of it. But not only are the hermeneutic problems limited to the literal reading process of code.

A general procedure of the construction process of a computer program needs to be detailed:

1. Interpretation and observation of the situation

2. Formalization and definition of its objects, properties and operations

3. Representation and implementation into computable code

First, the programmer needs to observe the given situation and deduce what is relevant for the problem at hand. These observations lead to the interpretation of the programmer which propagates through all other steps of the process. Secondly, the programmer goes on to formalize the relevant findings based on the previous observation and interpretation. Because of the rational symbolic nature of computation they must be defined as very specific rules. This definition consists of objects and the corresponding properties of the environment. Interconnecting them with operations which act upon the specified objects. Lastly, the programmer implements this formalization as a representation into code. This representation can look quite differently on the surface, depending on computer architecture and choice of programming language, but will always represent the underlying formalization as elaborated in the second stage.

Writing software is the act of formalizing the external world around us into abstract symbolic structures. It is the interpretation of the world into code, while cognitive representations are happening at every stage of it. The programmer enacts a very political action with writing software and is doing so with the limitations of the given background, being the pre-understanding and cognitive representations of the world. Winograd and Flores call this the “phenomenon of blindness”, building upon the terminology of the philosopher Martin Heidegger.

Heidegger combines Gadamer’s and Maturana’s thinking into his analysis of the world, profoundly questioning the rationalistic tradition and its core ontology. Just like Gadamer and Maturana, Heidegger rejected the separation of an objective physical reality and a subjective mental world or the possibility to describe an external “real world”. Important for us to understand are parts of Heidegger‘s thrownness (Dasein) and readiness-to-hand (Zuhandenheit).

Thrownness argues that we are, independent of our will, thrown into situations. Even if we decide to leave the situation or to not take part in it, they are inherently decisions and determined actions. Just like we cannot, not communicate; we cannot, not act. Plus, these inevitable actions are not directly perceivable to us. If we try to take a step back to reflect on the situation, we are blind to specific parts of it, which are therefore not perceivable and with that not part of the observation (Critchley, 2009). It is closely related to Maturana’s findings of direct representations. The same way the nervous system does not represent a direct representational state of a perceived situation, we can only reflect with our thrownness on it as such. There is no stable representation of a given situation. Looking at hermeneutics and its background of pre-understanding, we can move from the understanding of text, to the understanding of a situation and argue that a given situation cannot be perceived at without the context of interpretation and pre-understanding. Every representation which occurs in a situation, will have fundamentally different interpretations.

Closely related to the concept of thrownness Heidegger introduces another term called readiness-to-hand, with his in/famous example of hammering. He states that a person who is using a hammer, and is in the act of hammering, is actually not perceiving the hammer. For the person using it, the hammer is not identified as an object; the hammer doesn't exist as such. Rather it exists in the unconscious readiness-to-hand background. According to Heidegger the hammer itself becomes only perceivable as an actual object, when there is some kind of breaking down or unreadiness-to-hand occurring (Royle, 2018). Just like the hammer having a loose head, and with that becoming present-at-hand, is only perceivable as such in that situation, the inevitable thrownness of a situation puts us into a state of blindness. As a supposedly observer, we can see a given situation as an object and maybe reflect on its properties, but for the person being in the thrownness of the situation it is concealed and not identifiable as such. Thrownness and breaking down exists and shapes differently for every person

being part of a situation, being closely related to their pre-understanding and the hermeneutic circle.

With the concept of thrownness and readiness-to-hand, Heidegger inevitably describes the cognitive representation and hermeneutic fallacies and transfers them into everyday life.

Winograd and Flores concept of the "phenomenon of blindness", describes the previous ontological approaches and maps them into the environment of software development. Let us look at the translation of computation again to see, where this particular blindness takes places.

Interpretation and observation of the situation is the most critical and controversial stage from an ontological point of view. The observations and interpretations made in this step are very subjective to the programmer. They are based on the hermeneutic pre-understanding and are inherently not free from any prejudice. The programmer is inevitably engaged into Heidegger's thrownness, while trying to define a situation from an observer's standpoint, making it impossible to reflect fully on it. This stage serves as the utter most restricting as it decides about inclusion and exclusion of the consecutive computational equation. It determines who or what will be computed.

To acquire an awareness of a situation is, however, always a

> task of particular difficulty. The very idea of a situation means that we are not standing outside it and hence are unable to have any objective knowledge of it. We are always within the situation, and to throw light on it is a task that is never entirely completed. This is true also of the hermeneutic situation, i.e., the situation in which we find ourselves with regard to the tradition that we are trying to understand. The illumination of this situation—effective-historical reflection—can never be completely achieved, but this is not due to a lack of reflection, but lies in the essence of the historical being which is ours. To exist historically means that knowledge of oneself can never be complete. (Gadamer, 1975, p. 268)

Formalization and definition of its objects, properties and operations is the embodiment of the rationalistic tradition and computational thinking. While having a set interpretation present at hand (and present-at-hand), it needs to be even more simplified to be computable. At this stage further manipulations can happen, because of their rational nature they have to be formalized into. Not only the definition of an object with its properties serves as a highly complex task, the formalization limits the available possibilities by a great extend.

> In this way, computation does not merely govern our actions in the present, but constructs a future that best fits its

> parameters. That which is possible becomes that which is computable. That which is hard to quantify and difficult to model, that which has not been seen before or which is uncertain or ambiguous, is excluded from the field of possible futures. (Bridle, 2018, p. 44)

Representation and implementation into computable code is the stage that concludes the perpetual state of the interpretation. At this stage the formalized interpretations are being implemented as direct representations. While direct representations are a fallacy of our understanding of human cognition, they are very real in the space of software development. With them every software program serves a very concrete representation of the interpretations and formalizations of the programmer’s view on the given situation. It encapsulates the representations no matter what the implementation environment might be. During the implementation of the computable code the representation will happen, regardless of having any concrete concept of its executing environment. Code will execute. It only cares about its given representational state.

> In writing a computer program, the programmer is responsible for characterizing the task domain as a collection of objects, properties, and operations, and for formulating the task as a structure of goals in therms of these. Obviously, this is not a matter of free choice. The programmer acts

> within a context of language, culture, and previous understanding, both shared and personal. The program is forever limited to working within the world determined by the programmer's explicit articulation of possible objects, properties, and relations among them. It therefore embodies the blindness that goes with this articulation. (Winograd and Flores, 1987, p. 97)

With these concepts in mind, the programmer enacts a very disturbingly political action in writing software. It became apparent that the algorithmic bias is a very complex phenomenon and that it is a fundamental inevitable part of writing software, directly connected to the human writing it. It is crucial to move away from talking about objective or neutral algorithms and instead start expecting bias; instead of talking about facts, we have to start talking about interpretations.

MO–DELS

ARTICLE 06

Having looked at how bias lives in software per se, it is time to go back and join the discourse of algorithmic bias in machine learning. Exploring how to locate our previous findings in the field of machine learning. Machine learning is based on the concept of models, and with that it is closely related to the concept of cognitive representations. Fetching the english speaking wikipedia.com page with the search parameter "Model", a quick search for the term "representation" shows an allegedly consensus on the definition of a model. Representations are the essence of modelling. It doesn't matter in what discourse, field of study or industry it is defined; ultimately, a model is nothing more than an abstract representation of a given process. Whether it be about Amazon's fulfillment center supply chain, the bug fixes of REWE's new cashierless systems or the mere idea of your next meal. Models are everywhere and they don't always need to be in the form of computational code.

Being in the world of models and coming from cognitive representations, like we know them from Maturana and Heidegger, computer science started to treat the mind like a computational device. The field would talk about cognitive computation, different processors for vision and haptic, and different states of memory, declaring the early days of already successfully reinforced computational thinking. To figure out how humans understand and interact with computers, it seemed the most plausible idea, to also be able to deal with the incoming feedback by the human operator of the information system. While we know the difficulties of that idea, we are in the middle of the vice versa process, which of course has the same kind of problems. Nowadays we are trying to model the brain into the machine; generating computable models.

In the field of machine learning, the architecture of state-of-the-art models is based on the neural networks of our brains; at least on our current understanding of them. To fully understand how and where bias enters the process in machine learning, and to compare the differences to previously discussed forms of bias, we have to understand its technicalities and alter the translation of computation accordingly.

Just like an ordinary software algorithm, computational models feed from a specific set of inputs to generate a specific set of outputs based on rules. The main difference between an algorithm and a model is that the model's outputs are predictions. Predictions based on an internal generated algorithm based on the given input. To put it simply, a model is a replacement of the formalization, hence the rules, of an algorithm. We do this because defining specific formulations is hard. The idea is that we want to give the model some inputs and the correlating outputs; whereupon the model starts and learns the rules, so that when it is given some new never before seen inputs, it can create reasonable outputs, based on the fed in data by itself. But what are these inputs and outputs that define the model? In the field of so-called supervised learning, the programmer has to provide a dataset, resembling the inputs

and outputs that want to be achieved. The dataset that is fed into the model is crucial. It serves as training and testing data to shape the internal formalizations of the model.

Imagine a situation where the task would be as banal to write an algorithm that could distinguish cats from dogs. At first sight the problem might seem simple, but having a closer look at the rational formalized nature of algorithms, how do you define how a cat or a dog looks like and how do you define that input data? The first step would be to choose a medium, which would resemble cats and dogs, which the computer could interpret. Let's go with simple RGB images for now. They are handy, because they are easy to handle and already exist as formalized lists of bits and bytes aka. pixels, ready for the computer to read. So to re-define our algorithm: It is a classification of cat and dog images. Maybe not exactly what we wanted, maybe the initial solution would have been better to distinguish them in smell or sounds and not visual appearance.

Having that set we would start writing an image processing algorithm, which would take the pixels of an image as input and output a string of "cat" or "dog", to tell us the class of the image. The internal formalization of the interpretation of cats and dogs, would maybe look as the following: If, in this grid of 9x9 pixels, the contrast between darker and lighter pixels form a more triangular shape, it could possibly be a cat ear, but only if it is close to the other 9x9 grid, which resembles a part of a more circular looking shape, possibly being the start of a cats head. For the dog we would possibly go and look for a more upright posture or a longer snout with a more roundish looking nose. But what about dogs with triangular ears or cats that happen to walk upright? These classifications within the algorithm, being “small snout”, “triangular ear”, “long leg”, all representing a different arrangement of pixels, are called features; and it is almost impossible to get them right.

But this is exactly where supervised learning within artificial neural networks comes into place. We would not formalize these features ourselves, but let the model figure them out for us. We would train the model, based on the dataset we provide for it. So back to the images of cats and dogs. All we would have to do is to get a large amount of images and label them accordingly as “cat” or “dog”. The dataset would be a collection of data points consisting out of an image, also often referred to as the x value, and the corresponding label, being y. But where would we gather the images from? We could shoot some ourselves, to have full control over how the images would come out, but what exactly are we going for? Landscape or portrait, what kind of background, being still or being in action, inside our outside, more focused on the face or on the body statue? And most importantly, what kind of breeds would that include in our dataset? Making the pictures by ourselves gives us great control over the appearance of the picture, but we will probably only have a very limited amount of data to

work with. The common solution is the internet, serving as a vast collection of free to access data.

But what images will Google Image Search or Facebook really serve us? Several algorithms are already behind what the platforms are showing us. Adding another layer of potential bias would limit our process of gathering representational training data. All these decisions would lead to a different arrangement of pixels for the model to train upon, hence resulting into different features the model will predict with. But let’s pretend we managed to gather a relatively representational dataset of cats and dogs. Maybe we did it all by ourselves in the end, which by no means implies less bias, just more controllable bias. We would continue with the step of labeling our data, hence our pictures of cats and dogs. The potential problems with that are clear by now. Who or what is doing the labeling?

How specific do we want to be? Just two labels or maybe a label for different breeds? But what about cross-breeds then? Again pretending we made the right decisions for our specific use case, going for the sake of simplicity with just “cat” and “dog”, we are ready to so-called fit our model. Fitting the model is divided into a training and validation process. The model is training with a specific part of the dataset and after a given time it uses the other part of the dataset to validate its predictions. In the beginning the model will guess completely randomly if an incoming picture is a cat or a dog. At that point the model can calculate an error, because of our given supervised dataset. After each guess the model knows whether it was right or wrong and some validation epochs in, the model will start to recognize patterns in the data, hence our features we tried to manually describe before. Each feature describing a part of what it means to look like a cat or a dog.

The model keeps track of how much percentage a specific feature makes up to be an image of a cat or dog and after enough training, the model has a formalization of the specific features and their corresponding importance, representing our dataset. The model can be shown a completely new picture, which was not in the dataset before, and can predict with what percentage it recognizes a cat or dog. It can predict with what percentage the formalized feature list matches the features it can extract out of the given image. At this moment it is not easily comprehensible anymore what kind of features the model actually picked upon to be important for a given classification, but it is only working in the domain of our dataset that we gave it access to.

To compare it to the translation of computation, machine learning still embodies major parts of it. With the introduction of machine learning, a major part of the formalization stage might be replaced, but the inherent bias of the interpretation and representation are still binding and certainly present. The selection of data still resembles a very specific interpretation of the creator and raises completely new questions to be answered.

After the training process the model and its predictions are still set in stone, affecting the environment of implementation in the same way; and with that bias now camouflages itself even more behind the rationalistic nature of computation.

> Models are opinions embedded in mathematics. (O’Neil, 2017, p. 21)

To give a concrete example of how these models are already implemented in our world and don’t just affect our little side project of the classification of cats and dogs, I want to introduce the work of Cathy O’Neil. She worked as a math professor and left academia to work as a data scientist in finance and in e-commerce, actively being a part of the leading industries of computational models. Gaining increasingly insight into the nature of computational thinking and experiencing the 2008 financial crisis first hand, she realized the flawed parts of the system she was working in. In 2011 she started her blog mathbabe.org and in 2017 published her book “Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy” to spread the word about her research on the topic of algorithmic bias. Regarding this work, her book serves as an archive of flawed models, coming from first hand experiences and thorough research.

Let me walk you through the story of a model, which was designed to better Washington, D.C.'s underperforming schools to explain the direct impact on human life through modeled algorithms. Not to spoil: The model ended up a catastrophe, leading to undeserved job losses and excessive rise in fraud.

The process started with the false and overly rushed assumption that when students of a school are not performing well enough, apparently the teachers were not doing their job right. The newly hired chancellor of Washington's schools defined the plan of the new solution. With the model's objective set, being the evaluation of teachers, the teacher assessment software called IMPACT was developed. It would filter out the bottom two percent recieving the worst scores. Although the fifth-grade teacher Sarah Wysocki received overwhelmingly good feedback from the parents of the class, she was fired based on the evaluation of the algorithm. The new value-added method of the model would judge her effectiveness in teaching math and languages and the scores would be combined with the ratings of the community. Despite the positive ratings of the community, the score was not high enough for her to withdraw from the high influence of the model. IMPACT had such a high influence on the overall score, because the creators wanted to eliminate human prejudice. They argued that the community could be biased in favor of Wysocki, because of social connections or friends in high positions, which had nothing to do with her teaching skills.

> So Washington, like many other school systems, would minimize this human bias and pay more attention to scores based on hard results: achievement scores in math and reading. The numbers would speak clearly, district officials promised. They would be more fair. (O'Neil, 2017, p. 5)

This is a very typical phenomenon happening in the defense of computational models. The arguments always go in the direction of wanting to eliminate human bias with turning to "hard results". But the fallacy, as we know, is that the elimination of the human-in-the-loop just passes on the bias to the computational formalization, serving camouflaged bias in mathematics. Bias that is hidden inside the equations of the external model, only perceivable to the elite of the 21st century: mathematicians, data scientists and software engineers.

With probably similar thoughts in her mind, Wysocki wanted to know how the scores would actually be evaluated and how the value-added method was defined. She figured out, like it is so often the case, that the district had hired an external data science consultancy; in this case Princeton based Mathematica Policy Research. The value-added model would compare current test results against the results of the previous year taken by the same student. Like this, only comparing scores of the same students, the scores should be freed from any social status privilege and not be based on some general, global score average. While this approach definitely goes in the right direction of attempted fairness, the question of the teachers part in the computational equation still remains. How much of a potential score decline can you actually associate to the teacher? Wysocki expressed herself stating that

> There are so many factors that go into learning and teaching that it would be very difficult to measure them all. (O'Neil, 2017, p. 6)

And of course Wysocki completely got it right here. Because of the model's inherent nature of the computational platform and interpretations in the selection of the data, they are by definition generalizations and simplifications. If you are the exception of the dataset; the so-called noise that doesn't fit into the non-linearly separable function, you will get sorted out. As these generalized models decide more and more about our direct lives, they serve as a big threat to minorities and marginalized groups not fitting into generalizations.

As we cannot be certain on how big and diverse the dataset was that Mathematica Policy Research used for their model, a major problem could have been the amount of available data. Big Data companies like Facebook, Google, Amazon and co. models are trained on datasets with data points reaching into the 100 millions (Metz, 2012). Which of course also doesn't

guarantee any correctness of a model, but the amount of data is definitely a big part of it. Formalization take time and finding features is hard. A certain size of the dataset is a must to provide the model with enough variety to recognize patterns. In the environment of this specific model, we are dealing with classes of maximum thirty students, which results in thirty scores aka. thirty data points and this is just not enough to do serious analytics. Even if IMPACT was trained with data of several schools, the quality of the predictions are still highly questionable. In situations like this already existing models are often reused as so-called pre-trained models...

SELF-REIN-FORCE-MENT

ARTICLE 07

A pre-trained model is a model which was already trained and is ready to predict; a kind of plug-and-play situation. It is a common procedure when the internal classification is similar to the needed classification at hand. Especially when not much data is present this procedure is preferred. They serve as a counterfeit advance to have higher confidence values in the model's predictions right from the beginning of usage. The problem with pre-trained models is that they were trained and defined in a completely different environment than they are implemented. Just like in the representation and implementation stage of the translation of computation, without an update the model stays in the pre-trained state, never adapting to the new environment it is functioning in. Probably predicting on some data that has not much in common with the data that would be found in the situation at hand.

In the case of Washington's schools, the model never figured out if it was right or wrong. It never got to update its pre-trained state. All the model did was evaluating teachers without ever getting feedback on its predictions. Like it is common with many implemented models, the training process stopped at the most important step. Actually re-evaluating the models predictions with a human to oversight it (O'Neil, 2017). But in this case the model predicted away and all years long the predictions were executed. The model's will carried out and with that it was defining its own reality. A reality where the predicted score is the single source of truth.

As it turned out, in the example of the value-added model, people also understood this mechanism and started exploiting it. Investigations by the Washington Post and USA Today later found out that many of the standardized tests were fabricated. Not by the students, but by their teachers. The teachers knew that higher scores of their students would prevent them from being fired (O'Neil, 2017). Knowing that there are only a few variables defining their future, they started to play along in the reality of the model. The reality of the model leaked into ours, actively manipulating how people would behave. This scheme of computational thinking sits at the core of today's realities, highly influenced by how computers work. With the adaption of our thinking, we start to embody the model. Reinforcing its predictions into our lives. Onto our believes, onto our intentions and onto our behaviour. The model has the ability to self-reinforce itself through us.

Sarah Wysocki was sure that the fabricated scores were the reason for her misclassification. At the time of the low assigned score she was teaching first year students of a middle school. Her suspicion was that the scores of the last year's elementary school were inflated before, so when the student's new tests were fed into the value-added model, it would predict a bigger score gap. Unfortunately she was not coming close to the real truth why the model behaved the way it did. In her investigation the school district could only give very vague answers, mostly referring to the "hard results" of its outsourced solution (O'Neil, 2017). But for Mathematica Policy Research it was nothing more than a shipped product. The model was never updated to deal with the actual data in its implemented environment. It is common that anti-procedures of obvious algorithmic bias take place like this. The actual affected human being, most of the time being the exception in the dataset, is taking action against an algorithm, but in our information age it is not easy to appeal against a system based on "objective" math. As an emotional human being, there is no chance to appeal against a system grounded in the rationalistic tradition.

> The rationalistic orientation not only underlies both pure and applied science but is also regarded, perhaps because of the prestige and success that modern science enjoys, as the very paradigm of what it means to think and be intelligent. (Winograd and Flores, 1987, p. 16)

As long as the rationalistic tradition is the dominant way of how we think and treat each other, systems like IMPACT will not be held accountable for their actions. They will not even be looked at as a possible source of unfairness and discrimination. They will stay camouflaged and continue hiding behind the guardance of computational thinking.

Sarah Wysocki never got a justified answer that would explain the algorithm's decisions or why she lost her job (O'Neil, 2017).

To give another example of how models reinforce their beliefs into various fields of our societies, I want to display the in/famous phenomenon of predictive policing. Predictive policing programs emerged out of the Big Data field and are now, since several years, in great use all over the world (Friend, 2013). These crime prediction systems also mark machine learning as their main tool of success. The models use historical crime data as their input data and with that their predictions show where future crimes would most likely going to take place. Although reports show a decline in property crimes and in particular burglaries, the question of "historical crime" interpretations arises (O'Neil, 2017). What exactly is the model looking at? And what are the implementation consequences? Who acts upon the calculated predictions?

In the specific example of PredPol, the self-acclaimed market leader in predictive policing, product owners have a choice on what kind of crimes to focus on, hence what kind of crime data will fill the dataset. In a freshly installed system the software asks to also include so-called "Part 2" crimes, being vagrancy, aggressive panhandling and selling, and consuming small quantities of drugs. While the system mostly works effective with "Part 1" crimes, being more severe crimes like homicide, assault or burglary, the system's fairness starts to descent with the inclusion of "Part 2" crimes, led by

the "broken-windows" policing theory. The theory goes that environments, which look more careless or less maintained, hence the broken windows, would serve as a ground for more severe crimes. A house with a broken window would invite burglars. So people started to fix broken windows and to take more care of their environment. Unfortunately the movement eventually led to zero-tolerance campaigns, where police officers would stop and arrest low-level crimes, filling U.S. prisons with numerous people convicted with victimless crimes (O'Neil, 2017).

Back to PredPol, the software works with the Google Maps web interface, displaying its predictions as 150 m2 boxes with red borders signaling high-risk areas. These areas resemble the area where a crime is most likely going to happen, so police officers are instructed to patrol these areas more often. With the "Part 2" crimes feature activated, neighborhoods of low-level crimes are becoming the hot-spots for patrolling. So when the police officer visits the predicted area and sees teenagers sharing a joint on the street or any other low-level, victimless crime, the officer will stop them. With that PredPol's cloud would spin up and start evaluating the new captured data, which of course would be evaluated as a success, because the new location data point of the new crime would match within the predicted red bordered box. So the model would update its "patrol heat maps", sending more patrols to the same area again leading to more reported crimes of that neighbourhood, resembling another vicious self-reinforcement mechanism.

> The policing itself spawns new data, which justifies more policing. (O'Neil, 2017, p. 87)

But what about the question of the dataset? Where is the crime data coming from and on what interpretations are these models build upon? In the example of PredPol, the model is fed with the data points from the specific agency's records management system (RMS). This system is a vast collection aka. database of previously filed crimes of the police department or district. A collection which resembles the numerous interpretations of police officer from the past. Our models which are predicting and enforcing the future we will live in are based on interpretations from the past. And we all know that there are major parts of the history of our societies we wouldn't like to repeat. The racist and discriminating past of police policing is one of them.

SAME DATA (SINGLE SOURCE OF TRUTH)

ARTICLE 08

But when did we start to predict the future based on computational systems? We always wanted to look into the future and computational systems looked like promising help. In 1922, computers as we know them today didn't exist, but their architecture and computational power were already dreamed of. In his writing "Weather Prediction by Numerical Process" the mathematician Lewis Fry Richardson made first attempts to compute the future. He did so by dividing the world into squares and data points, trying to predict the weather with pen and paper. His six-hour forecast would take six weeks to complete. But to look into the future of the weather, calculations would have to be faster than the actual forecasting period. Eventually starting a race against time, trying to predict the future faster than it would arrive.

But with models from today in mind, did we really just want to look into the future? Or did we strive for control? Computational systems have that control now, constructing future through the use of predictions, which are based on the past. With the use of historical data we are reinforcing the past, leaving no room for change. We are actively assuming that the past will linearly continue to be the future; without regarding that the future might have a different course than the past and without regarding that the future might have different values than the past. Computation blurred the lines between past and future, creating the most vicious feedback loop in the age of computation; the way things are will be the way things will be.

> Because mathematical models, by their nature, are based on the past, and on the assumption that patterns will repeat. (O'Neil, 2017, p. 38)

> In this way, computation does not merely govern our actions in the present, but constructs a future that best fits its parameters. That which is possible becomes that which is computable. (Bridle, 2018, p. 44)

And is this really what we should be striving for? In the example of PredPol and predictive policing the only computable data available was based on the previously cited crimes, which resembled a vast collection out of interpretations and observations from various police officers on patrol. And these observations were anything but fair and unbiased. Our past is inherently discriminating and we don't want that these discriminating patterns are reinforced into our future.

Staying with the example of predictive policing, a majority of the collected patrolling data was based on the so-called stop-and-frisk method. "Stop, question, and frisk" was one one the major anti crime policies of the New York City Police Department. The concept was the following: the more people you stop, the more crimes you can prevent. The idea was simple: police officers stopped anyone that looked suspicious to them. They asked for their ID and frisked them. The number of stops went up by 600 percent. And with the statistics showing

a decline in crime, it was called a success. But an efficient system is not always fair. The policy was later described as uneven policing, pushing more minorities into the criminal justice system. The administration, the method was running under, was sued by the New York Civil Liberties Union, charging the stop-and-frisk policy as being racist (O'Neil, 2017).

Taking a look at the dataset of the policy, which is publicly available on NYPD's website. For the year 2018, the dataset lists 11.009 stops in total, from which only 1.074 were of white people (New York City Police Department, 2019).

Continuing with the racist history of failed systems, let's take a look at computer vision systems. In the movie industry around the 1960s Kodak films were widely used and resembled the industry standard at that time. Unfortunately the films were designed for people of white skin color, because the model posing for the Kodak test cards, which were used to calibrate the color films, happened to be white. The films were unusable for working with darker skinned people. The pictures would always turn out to be under-exposed. The most ironic part is that Kodak didn't change their calibration methods. Kodak just didn't update their model to work with darker tones. But when complaints, not being able to photograph chocolate or dark horses, were filed, only then Kodak reacted (Pater, 2016). Another rather troublesome example is when Google's Photo app image classification algorithm failed and labeled two black people as "Gorillas" (@jackyalcine, 2015). In June 2015 @jackyalcine tweeted about the incident, but all what Google did was to remove gorillas altogether from its model (Simonite, 2018). More precisely, until this day (05.03.2019) Google still blocks all search queries for the tags “gorilla”, “chimp”, “chimpanzee”, or “monkey” in its Photo app. Just because the pretrained model is not being updated; since almost 4 years now. Behind the app sits most probably the wide spread ImageNet dataset. ImageNet is an academic dataset, which provides a numerous set of images according to the labels of a database called WordNet. For each “synonym set” of WordNet, ImageNet tries to provide an average of 1000 images. Browsing the explorer on the dataset’s website, taking a closer look at the node “People”, quickly shows the deficiency of black people, which is not representational and most probably leading to the misclassification of Google’s Photo app (ImageNet, 2019).

The phenomenon of automation bias shows a repeating pattern, which can be observed in the use of computational systems. Again tightly connected to computational thinking, automation bias describes our unconditional love and trust in computational systems. In the national park in Death Valley, rangers came up with the term “Death by GPS”, because the event of people following the blue route until their death was occurring again and again. It might seem implausible, but the trust we have in our everyday algorithms, is astonishingly high. Another incident happened when a tourist group drove

their car into a lake, because the navigation system told them too (Bridle, 2018).

Whether it is the dataset, the model, the app, or the blue route, we treat every stage of the design of computational systems as the single source of truth, resulting in an inevitable cascading waterfall of unpredictable bias.

> The problem is not only in the semantic bias of the data set, but also in the design of the algorithm that treats the data as unbiased fact, and finally in the users of the computer program who believe in its scientific objectivity. (Cramer, 2019, p. 33)

We build these highly optimized systems, but the underlying foundation is again and again disregarded. The same is happening in the discourse of algorithmic bias and machine learning. Most of the biases are not a new phenomenon of machine learning, they are inherently rooted in software development. The data that we feed our systems with is not looked at enough, and as we know from the translation of computation its the most crucial step, where all interpretations and observations take place.

MISSING VARIAB-LES AKA. PROXIES

ARTICLE 09

The story of the missing variable is an important one. As we know by now, the interpretation and observation question of what to include in a software model is crucial. Because of machine learning's nature to be divided into several parts, being data mining, dataset normalization, dataset implementation, model definition etc., the significance of it is increased by a great deal. Interpretations are happening at every stage of the process, making it an opaque assemblage of potential biases. It is not only the bias that is actively embodied in the dataset, but also the exclusion of information, which shapes the bias. A transparent documentation of inclusion and exclusion becomes necessary.

With that in mind it is necessary to look at the variables which were left out of the process. Although a data point was not included in a dataset, it can be of great significance for the model's predictions. So in a sense a missing variable is nevertheless included into the model. The data point still exists in the real world, whether it was included in the model or not, and with that the model will have implications on it. And most of the times it is exactly that variable, which was chosen to be left out of the computational equation, that shows the most significant implications in the case of a breakdown.

In the case of Google's Photo app, this is exactly what happened. The representation of people in the model's dataset was not diverse enough. Black people were chosen to be the missing variable. And of course they were the most affected ones by the incident. The unfortunate part from Google was that they didn't fix the diversity of the dataset, but they went with deleting the misclassified labels, which just led to more exclusion and bias in the model.

While this explanation seems quite obvious, it is not that simple from the software developer's point of view. It is always easy to see what is in front of you, hence what is being included into a system. Being decidedly critical with the situation at hand, to detect important variables which are being excluded from the observation and implementation, is hard. But bias will exist, whether it is intentional or unintentional.

In statistics the intentional implementation of stand-in data, hence to substitute missing data points, are so-called proxies (O'Neil, 2017). In the development of models the actual interesting data is often missing, so the creators have to find a substitute for them. Other more accessible data points are looked for, which could resemble the actual desired variable, and are used to describe the new artificially created proxy. Proxies serve as a difficulty for the correctness and fairness of a model, since they add another layer of interpretation to the creation of the computational system. Proxies are by their very definition a short-cut or the easy way out, since they serve as a simplification of a part of the world which could not be measured or defined easily. And later in the process of evaluating the model or updating it, the proxy variables are hard to spot

and redefine, since they are treated as usual variables in the model's predictions. The model's predictions which are based on an artificial extrapolation of the world around us.

Looking back at the example of Sarah Wysocki, it is clear what damage the implemented proxies did. A whole ability to teach students was reduced to a single score. The creators of the model just didn't have the resources to explicitly define what it means to be a great teacher. They didn't conduct much research in the field of education or development of knowledge, because the already given computable data of the underlying value-added model was serving as an easily exploitable data point. Which ironically, was actually exploited by the people affected by the model. Just like the creators exploited a simplification of our reality, this simplification was identified by the outside and was used to trick the system. Not only are proxies a great danger to the actors being simplified, they also serve as an alarming attack surface to exploit the model. They are shortcuts in both ways.

The same happened to the racist PredPol model. Looking back at PredPol's predictive policing model, the founder of the company, Jeffrey Brantingham, told O'Neil that

> [...] the model is blind to race and ethnicity. [...] PredPol doesn't focus on the individual. Instead, it targets geography. (O'Neil, 2017, p. 86)

Which PredPol's website also confirms. They state that they only use three data points which are the type of crime, crime location, and the crime's date and time (PredPol, 2019). But how comes that the inspected dataset from NYPD's stop-and-frisk policy, which is highly correlated to PredPol's model, still shows a bias in favor of white people? This happens because the location data point is working as a perfectly fine proxy for race. In our segregated cities, your location is saying much about your ethnicity and social background.

To give another example of how proxies are used in a production environment, I want to introduce another example from Cathy O'Neil's research. In 1983 the news magazine "U.S. News & World Report" started to rank several major colleges and universities from the U.S and would publish its findings as "U.S. News Best Colleges Rankings". In the beginning the list was solely based on an internal survey send out to university presidents. But as the complaints entered, the magazine needed to come up with a more sophisticated model. But again confronted with the fundamental question, on which data will the model be based on? They decided to stay with the survey and give it a 25 percent cut of the whole evaluation. The remaining 75 percent would be the model trained on "educational excellence". Of course "educational excellence" represented as a vast collection of proxies, which were debatable to represent the desired goal. Again the fundamental embodiment of the

rationalistic tradition did its duty and U.S. News later defined "educational excellence" with SAT scores, student-teacher ratios, acceptance rates, drop-out rates, and number of alumni who would actively contribute money. The complex reality was once again simplified to a bunch of computable data points (O'Neil, 2017).

Referring back to the importance of the missing variable, this particular model has a great confession to make: tuition fees were not included in the prediction of the model. Ever since the ranking went public tuition fees skyrocketed. Leaving many students and families with depts in unimaginable amounts.

Nevertheless the ranking became a national standard and with that a much bigger problem arose. Just like it is often the case, the model reinforced itself. Because of the wide acceptance, the ranking became the new objective for all colleges and universities. Everyone wanted to be at the top of the list, because that‘s where everybody was looking at. And as we know from the nature of proxies, once they are implemented and out in the open, they will be exploited to practice deceit. And this is exactly what happened. The ranking turned into a race of who can trick the algorithm the best, where the value of education just didn‘t matter anymore.

But there is no need for us humans to put these shortcuts into our systems, with machine learning in the tool belt we are set and ready to interpolate. Essentially machine learning is a proxy producing gambling machine. With pattern recognition, we never know what we are going to get and how the importance weights of the various features will look like.

Looking back at our cat and dog classification project, I want to add another important step in the creation of the dataset. While preparing the dataset we would have to pay attention to not create wrong correlations in the dataset. Let me explain what I mean by that: when taking the pictures of the cats and dogs, we would have to normalize them in a way, that the model would actually pick upon the features we want it to pick upon, meaning the visual difference between a cat and a dog; and nothing more. If we would decide to place all cats in front of an orange background and all dogs in front of a purple background, the model would essentially also assign the feature of a lot of orange pixels to cats and the feature of a lot of purple ones to dogs. So when we would place a dog in front of an orange background, the model would most probably, predict it to be a cat. While training the model, the classification can pick up very fine features, and there is no guarantee that it would be a feature which is in correlation with our initial goal, meaning there is no guarantee if the recognized pattern is a proxy or not; making pattern recognition a proxy producing gambling machine.

> Machine learning does not distinguish between correlations that are causally meaningful and ones that are incidental. (Agüera y Arcas, Mitchell and Todorov, 2017)

The same is what probably happened to the model of the paper "Automated Inference on Criminality Using Face Images" of Xiaolin Wu and Xi Zhang, which was extensively analysed in the great article "Physiognomy's New Clothes" by Blaise Agüera y Arcas, Margaret Mitchell and Alexander Todorov. Wu and Zhang's paper states that they can predict

> the likelihood that a person is a convicted criminal with nearly 90% accuracy using nothing but a driver's license-style face photo. (Agüera y Arcas, Mitchell and Todorov, 2017)

When having a closer look at the sample of the dataset provided in the paper, it doesn‘t need much to spot a clear classification feature, which is not part of the face. All three people labeled as "non-criminal ID" are wearing a white collar shirt, being a way too easy catch for our proxy gambling machine.

HARD-CODED

ARTICLE 10

Hard-coded is an often referred to term in software development. It describes a variable, expression or statement that is absolute. Absolute meaning, absolute to the context the program is written in, and with that not relational or flexible to the environment. A hard-coded variable is set in the initial programming environment, and would break it, if the environment would be changed. It cannot adapt to a new implementation environment.

With the constant representation of the external world in writing software, human bias was so to say hard-coded into the program. The programmer's prejudice and view of the world was directly hard-coded in it; it would be very explicit and concrete, and wouldn't adapt to any new situations. If a programmer wanted to include a specific property of an object but not another one, it would be absolute, no matter the environment the software is going to execute in.

One could argue now that through the implementation of machine learning, the before defined absolute biases are not hard-coded anymore. The argument would be that they emerge from the dataset, which was used to train the model, and with that they would emerge as generalized patterns of biased data. While this argument holds true within the supervised training process of a model, looking at the implemented and fitted model, the absolute nature of the dataset uncovers. The generalized patterns of the neural network camouflage the absolute data points of the dataset as being relational variables.

After the training process is completed, the predictions of the model are absolute: a given x will always result in the same y, hence a given input will always result in the same output. With that in mind, machine learning models are also hard-coded, being inflexible and absolute. Just like the traditional software program, the model is curated by absolute statements of the programmer, being hard-coded into the context it was created in. Bias is and will always be hard-coded; hard-coded by the human creating the piece of software.

Having looked at the emergence of algorithmic bias and how it manifests itself in software development, one coherent theme becomes apparent: there is no such thing as algorithmic bias per se; the occurrences of bias are diverse and vary in form and implementation. While the bias always finds its origin in the human, through computational systems it abstracted itself away from it and continues to live in different layers of our complex systems.

The term "algorithmic bias" or "biased algorithm" is misleading to the public. It reinforces the thought that the algorithm is biased itself. Just like in the whole process of its creation, it once again shifts the responsibility away from the human being. While the bias inherently becomes a property of the algorithm itself, the term implies that it originated within the algorithm and deflects from the human creators building the

systems.

Bias is inevitable and the hard part is not getting rid of it, but spotting where it enters our complex computational systems and to decipher the parts of the bias which are discriminating and are actively harming people.

To better define and locate the bias, I would propose classifying the term "algorithmic bias" into three main problems, which resemble the underlying domains where bias enforces in software. I see the bias in software development as an ontological problem, a technical problem and a reinforced problem. However, the separation doesn‘t imply that the domains work apart from each other. They are interdependent. In its manifestation, bias always exists as a hybrid out of these three problems/domains:

The *ontological domain* sits at the core of writing software. It juxtaposes computation with the limitations and preconceptions of human perception. With the work of Gadamer, Maturana and Heidegger it describes the fallacies of interpretation and representation. Their concepts of the hermeneutic circle, the fallacy of instructive interaction and thrownness play a vital role within the context of the inevitable human bias while observing and formulating the physical and relational world around us.

The *technical domain* defines computation's origin within the rationalistic tradition. It states the fundamental and binding technical bias of computational systems. Code itself is not neutral nor objective. It exists as a highly rational medium, which creates and favors a specific way of thinking and way of being. Within the advanced rationalistic tradition of today, rational thinking is not only favored, but the only option specified by the computational systems. The domain questions the deeply implemented rational fundament software is compiling onto.

The *reinforced domain* is the layer which functions as a catalyst to the underlying problems and limitations. It commonly consists of abstracted computational concepts based on existing computational systems, reinforcing the rationalistic tradition and practising highly complex hermeneutic methods. Through the reinforcement the previous domains gain a protecting layer, making them more opaque, less accessible and less changeable. The technologies used to reinforce the previous domains, develop means on their own to further stimulate the needs of the rationalistic tradition and its consequences. Because of the cascading reinforcement the direct implications of computation become increasingly incomprehensible.

Within these three domains bias doesn't exists as a static definition or limitation per se; it exists as a dynamic process, which gradually emerged within the computational systems;

historically and practically speaking. With the ever cascading and accumulating interpretations, formulations and representations in software development, bias is not only a single property with destructive consequences, but it can also be seen as a temporal course and thus be defined as a process. A process which needs to be acknowledged to reflect onto our ontology to increasingly understand who we are and how limiting we perceive the world around us. A process which needs to be constantly updated to synchronize the world with the created rational abstractions. And a process which needs to be transparent to disclaim the decision making of inclusion and exclusion to the computational affected environment.

CLASSIFYING A SPECTRUM AND THE PATTERN RE-COGNITION OF HUMAN BIAS

TECHNICAL DOCUMENTATION

The accompanying practical work is the creation of a conceptual dataset. It finds its purpose in communicating and exploring the concept of classifying a spectrum. It is juxtaposing the unacknowledged and unrecognized spectrums occuring within the interpretations and representations of software development with the widely acknowledged visible electromagnetic spectrum. Within the act of classifying a spectrum the work reflects on the rationalistic practises and techniques it takes to make our complex world quantifiable.

Sticking to the rather materialist view and how it is giving the tool itself the property and power to shape human behaviour and thinking, my work finds appeal in this approach, giving the tool the possibility to change its way of being used. The definition of a tool is pre-defining the way it is used and understood. The tool's values are hard-coded into its creation.
With that in mind I redefine the technique of machine learning to the pattern recognition of human bias. Changing the initial perspective and incentive of machine learning takes the materialist approach and redefines machine learning's function.

1. From hard-coded human bias to generalized patterns of biased data

2. Normalized data with human bias into pattern recognition

3. Machine learning as pattern recognition of human bias

These two approaches use the technological practical juxtaposition of machine learning to learn and reflect onto our limitations and reinforcements within the previously stated domains. The transduction throughout the three domains give us the opportunity to explore human bias within and through machine learning.

Within this technical documentation I give insight in my constant subjective decisions that occurred while creating the computational environment around the practical work.

The work is a new media installation which finds its origin within web development. The starting point of the installation is the interface which is used to define and gather the dataset. The interface is a simple website which consist out of three main parts: classification, dataset and writing.

The classification starts with the generation of a random color. Within the function `setColor()`, three floored numbers between `0` and `255` are generated. They resemble the three channels of the RGB color model, being red, green and blue, which is a widely used additive color model to generate colors in electronic systems.

```
const r = Math.floor(Math.random() * 256)
```

```
const g = Math.floor(Math.random() * 256)
const b = Math.floor(Math.random() * 256)
```

From these three variables a new object `color` is being created to define a string which is then used to display the color via the Cascading Style Sheet property `background`. The raw string of the red, green and blue values is also synced to an input field below the generated color to work as the input for the predictions of the trained model.

```
this.color = {
  r: r,
  g: g,
  b: b
}
```

```
this.ui.rgb = `${ this.color.r }, ${ this.color.g },
${ this.color.b }`
```

Below the generated color and the RGB string input field, is the list of possible color labels to pick from. The user visiting the website classifies the generated color, with assigning one of the labels to it. When one of the color labels is clicked, the generated color value with the according color label are gathered and prepared to be stored in the dataset. After a simple validation, which handles possible misuse of the input fields of labels, the function `sendColor()` sends out the new data point with instructions to the backend of the application.

```
const msg = {
  do: ‘insert-color‘,
  data: {
    label: label,
    color: this.color
  },
  client: this.client
}
```

Looking at the object from down to top, the `client` key is a unique identifier which is automatically generated on visiting the website. The `data` key holds another object with the generated data points including the label with the according values. The `color` key resembles the x value, while the `label` key stand for the y value, which are later being used for the supervised training process. The `do` key of the object gives the receiving server the instructions on how to handle the incoming data. In this case the string `insert-color` tells the server that it is a newly generated data point which should be inserted into the dataset. A timestamp and a unique id are added to the entry in the database.

```
{
  “data“: {
    “label“: “violet“,
    “color“: {
      “r“: 203,
      “g“: 85,
      “b“: 184
    }
  },
  “client“: “62e77608efb71“,
  “timestamp“: 1545934945977,
  “_id“: “b5GjklTlfQjtaQzt“
}
```

If a generated color cannot be described or classified within the given labels, the user has the ability to add a new label via a second input field, which is then available to further classifications. With that the dataset’s diversity of y values is flexible and updateable.

The second area of the website gives the ability to interact with the application programming interface of the curated dataset. It gives insight in the gathered data points and visualizes them accordingly. After selecting a color label from the dropdown, the server selects all matching data points and sends them back to the client. The website then renders the received data points, with their corresponding color values, on the screen, giving a visual representation of the classified data. Each data point is clickable, revealing the underlying entry of the dataset as a direct copy from the database. The end points of the application programming interface are open and completely accessible to the public.

The writing area shows paragraphs of text consisting out of excerpts of this writing to give the project more context and to explain the theoretical background. Before the paragraphs are rendered onto the screen, the excerpts are manipulated by a regular expression script to properly display literature references and to enable basic syntax highlighting.

On the server-side the machine learning model is being trained in variable intervals. The standard value to retrain the model is every 100 new data points. Everyday at 3:00 AM the model is saved in an archive directory to document the process of the training process. The intervaled retraining guarantees the frequent synchronization between the model and its environment. The model itself is build with Google's open source machine learning library Tensorflow.js. It is a standard sequential neural network with three layers. The network's architecture consisting out of the first layer having an input shape of three to resemble the three values of a color data point, the output layer adapting to the current amount of color labels in the da-

taset and the hidden layer varying its amount of units to give the network enough space to pick upon the features.

```
const model = tf.sequential()

const hidden = tf.layers.dense({
  units: 16,
  activation: ‘sigmoid‘,
  inputShape: [3]
})

const output = tf.layers.dense({
  units: 6,
  activation: ‘softmax‘
})
```

Within the website and on the initial page load the latest model is downloaded. The red, green, blue value input field is editable and serves as the input value of the model. When the input field is focused the model extracts the three values and displays the predicted label based on them.

The installation setting consist out of an input device and several output devices. The input device is an Apple iMac G3 displaying the described interface to gather new data points during the exhibition. The almost 20 year old computer should resemble the manifestation of the technical domain, which also includes limiting interfaces and input methods. The output devices are recycled computer monitors powered by Raspberry Pis. They show a comparison of datasets and give contextual insights in technical process and the underlying theoretical work. The installation function as a live classification of human bias.

The full source code can be found on `https://github.com/ccccccccccccccccnrd/translation-of-computation`

@jackyalcine. (2015). Google Photos, Y'all fucked up. My friend's not a gorilla. Twitter. Retrieved from https://twitter.com/jackyalcine/status/615329515909156865 on March 25, 2019.

Agüera y Arcas, B., Mitchell, M. and Todorov A. Physiognomy's New Clothes. Medium. Retrieved from https://medium.com/@blaisea/physiognomys-new-clothes-f2d4b59fdd6a on March 25, 2019.

Bridle, J. (2018). New Dark Age: Technology and the End of the Future. London: Verso.

Cramer, F. (2019). Pattern Discrimination: Craptularity Hermeneutics. Lüneburg: meson press.

Critchley, S. (2009). Being and Time, part 4: Thrown into this world. The Guardian Religion. Retrieved from https://www.theguardian.com/commentisfree/belief/2009/jun/29/religion-philosophy on March 25, 2019.

Friend, Z. (2013). Predictive Policing: Using Technology to Reduce Crime. FBI Law Enforcement Bulletin. Retrieved from https://leb.fbi.gov/articles/featured-articles/predictive-policing-using-technology-to-reduce-crime on March 25, 2019.

ImageNet. (2019). ImageNet Explore. ImageNet Project. Retrieved from http://image-net.org/explore on March 25, 2019.

Lipton, Z. (2016). The Foundations of Algorithmic Bias. Approximately Correct. Retrieved from http://approximatelycorrect.com/2016/11/07/the-foundations-of-algorithmic-bias/ on March 25, 2019.

Maturana, H. R., Lettvin, J. Y., McCulloch, W. S., and Pitts, W. H. (1960). Anatomy and Physiology of Vision in the Frog. Cambridge: The MIT Press.

Metz, C. (2012). Facebook Tackles (Really) Big Data With "Project Prism". Wired Business. Retrieved from https://www.wired.com/2012/08/facebook-prism/ on March 25, 2019.

New York City Police Department. (2019). Stop, Question and Frisk Data. NYPD Stats. Retrieved from https://www1.nyc.gov/site/nypd/stats/reports-analysis/stopfrisk.page on March 25, 2019.

O'Neil, C. (2017). Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. London: Penguin.

Pater, R. (2016). The Politics of Design: A (Not So) Global Manual for Visual Communication. Amsterdam: BIS Publishers.

PredPol. (2019). Predictive Policing: Guidance on Where and When to Patrol. PredPol. Retrieved from https://www.predpol.com/law-enforcement/ on March 25, 2019.

Richardson, L. F. (2010). Weather Prediction by Numerical Process. London: Forgotten Books

Royle, A. (2018). Heidegger's Ways of Being. Philosophy Now. Retrieved from https://philosophynow.org/issues/125/Heideggers_Ways_of_Being on March 25, 2019.

Sangosanya, W., Belton, D. and Bigwood, R. (2005). Basic Gates and Functions. Digital Logic. Retrieved from http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/ on March 25, 2019.

Simonite, T. (2018). When It Comes to Gorillas, Google Photos Remains Blind. Wired Business. Retrieved from https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/ on March 25, 2019.

Winograd, T. and Flores F. (1987). Understanding Computers and Cognition: A New Foundation for Design. Norwood: Ablex Corporation.

VERSICHERUNG

Hiermit versichere ich, dass ich die Arbeit selbstständig angefertigt habe und keine anderen als die angegebenen Quellen und Hilfsmittel genutzt habe. Zitate habe ich als solche kenntlich gemacht.

Köln, 26. März 2019