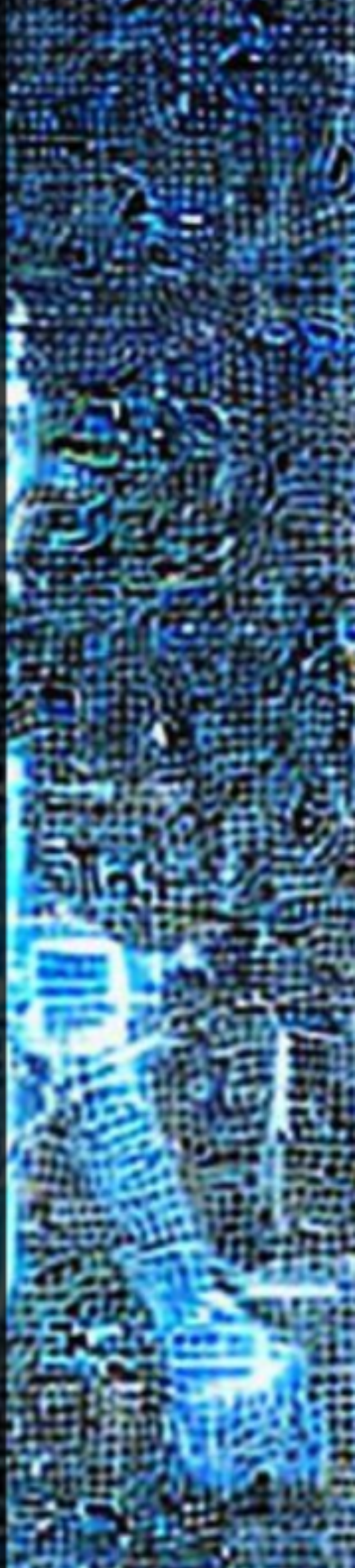
First Edition

Full Self-Driving, Skynet

and other AI Myths

Brad Flaugher

Modern decision making with deep machine learning models



Full Self-Driving, Skynet and Other Artificial Intelligence Myths

Modern decision making with deep machine learning models

Brad Flaugher

January 25, 2023

We have now accumulated sufficient evidence to see that whatever language the central nervous system is using, it is characterized by less logical and arithmetical depth than what we are normally used to.
– John von Neumann [1]

Preface

This book is an attempt to organize my brain on paper. I have been thinking about decision making with computers, and the computer's relationship to the brain for too long now, and before I get dementia (I'm 35) I would like to put a stake in the ground and tell everyone where they can stick their thoughts about the future.

I am simultaneously very hopeful for the future and horrified by the way that people talk about it. I think many so-called "Data Scientists" are full of shit when they talk, and some of them know it but would like to keep their jobs. I think that many investors are snowed by fancy language and toy demos that they extrapolate into a magical future. Users of digital products are woefully ignorant about how the products work and how they fit into a larger ecosystem. I'm amused, annoyed and would like to clear things up for everyone, including myself.

I am a self-employed programmer and financially secure, these are my credentials. I have no corporate master and I am not raising money to create the future of AI, so have a limited bias towards AI boosterism. I write programs close enough to the bleeding-edge that I can see the present as fast as almost anyone. Because I am simultaneously a contractor, investor and teacher I get the opportunity to see many projects, and I hope I can effectively comment on some common misconceptions and errors in thinking I see in many businesspeople, developers and investors.

So let's go on this journey together, I'll try and explain to you (with words, code and examples) how modern AI works, and when you should be scared (self-driving cars without the appropriate infrastucture) or when you should not be scared (please shut up about Skynet becoming self-aware). I'll also try do case studies in as many arenas that I think are interesting. If you have notes for me you can reach me at brad@bradflaugher.com.

Please skim this book, please skip around and read some stories. I think the first three chapters are important for everyone to know. The rest of the stories can be chosen a la carte. Enjoy.

Brad Flaugher

Contents

History: The End of Good Old-Fashioned Artificial Intelligence

1

"Programming will be obsolete. I believe the conventional idea of "writing a program" is headed for extinction, and indeed, for all but very specialized applications, most software, as we know it, will be replaced by AI systems that are trained rather than programmed. In situations where one needs a "simple" program (after all, not everything should require a model of hundreds of billions of parameters running on a cluster of GPUs), those programs will, themselves, be generated by an AI rather than coded by hand." Matt Welsh, 2023 [2]

1.1 The AI Textbook in 1997

Dr. Elaine Rich's textbook on Artificial Intelligence, published in the 1980s, was a groundbreaking work that helped to establish many of the foundational concepts and techniques in the field of AI. However, the rapid advancements in AI over the past few decades have led to many of the chapters in this textbook becoming obsolete.

One of the main reasons for this is the prevalence of deep learning, big data, and large-scale statistical models in modern AI. These techniques have largely replaced the symbolic, rule-based approach to AI that was emphasized in the textbook, making many of the chapters on knowledge representation and expert systems less relevant.

Additionally, the explosion of data and the availability of powerful computing resources have made it possible to apply machine learning techniques at a scale that was previously unimaginable. This has led to the development of highly effective machine learning models that can handle complex tasks such as image and speech recognition with a high degree of accuracy, making many of the chapters on simpler machine learning techniques such as decision trees[1] and linear regression less relevant. [3] [2]

We'll discuss this history and a few examples from the "early days" of AI to help us understand where we are headed. We'll start with machine translation, then discuss chess and finally neural networks, which will be the focus of the rest of this book.

1: Although mathematically, Neural Networks are Decision Trees

[3]: Rich et al. (2009), *Artificial Intelligence*

2: the book is now in its third edition and unlikely to be updated as Dr. Rich as retired utexas.edu

1.2 Does AI Need to Know Grammar to Translate?

Noam Chomsky is a linguist and philosopher who has made significant contributions to the field of linguistics with his theory of universal grammar. Chomsky believes that all human languages share a common underlying structure, and that this structure is innate to humans. He proposes that this innate structure is the result of a "language acquisition device" present in the human brain, which allows us to learn and produce language. Chomsky

also argues that the structure of language is largely independent of its content, and that the ability to produce and understand language is a fundamental aspect of human nature. His theory has been influential in the field of linguistics and has sparked much debate and research on the nature of language and its relationship to the human mind.

For English speakers or anyone who has learned English as a second language you'll have many examples of special cases, irregular verbs, bad english and former street slang that became good and proper over time. For programmers this is a nightmare, how can we codify human knowledge in a timely fashion? If we tried to write the rules of the english language in code (which many have tried to do) the rules themselves might change before we were finished writing them.

Explicitly translating languages through code is a difficult task because it requires a thorough understanding of the grammar, vocabulary, and syntax of both languages, as well as the nuances and subtleties of their respective cultures[3]. Simply coding rules for how to translate words or phrases from one language to another is not sufficient, as there are often multiple valid translations for a given phrase depending on the context in which it is used.

3: For programmers this is a nightmare, how can we codify human knowledge in a timely fashion? If we tried to write the rules of the English language in code (which many have tried to do) the rules themselves might change before we were finished writing them.

A more effective approach to translation is to use statistical techniques that rely on a large corpus of translated data, such as Canadian laws[4]. This type of data-driven approach involves training a machine learning model on a large dataset of translations, allowing it to learn the patterns and relationships between the languages. The model can then use this knowledge to make educated translations of new phrases or sentences, taking into account the context in which they are used.

4: They're in French AND English, which is useful data that we can use to correllate phrases and transform English to French and vice-versa.

While this approach is not perfect, it has proven to be highly effective in machine translation and can produce accurate translations even for languages that are very different from each other. The use of a large dataset of translations also allows the model to learn from the mistakes and variations present in real-world translations, further improving its accuracy.

1.3 Explicit Rules and Codified Human Knowledge

When we "teach" a computer to perform a task by explicitly writing down all of the rules of that task, we are really codifying human understanding.[5] When we codify human understanding we write down every rule that we know explicitly. For small tasks we can do this with 100 percent accuracy, and only minor headache on the part of the sofware developer.

5: Programming this way makes some software development totally boring, I almost switched my major in college to math after considering what a life would look like manually writing rules for handling "edge cases" for the rest of my natural life.

For example, let's write a boring function to tell you the number of days for a given month.

```
def days_in_month(year, month):
  if month in [1, 3, 5, 7, 8, 10, 12]:
    return 31
  elif month in [4, 6, 9, 11]:
    return 30
  elif month == 2:
    if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
      return 29
    else:
      return 28
  else:
    return "Invalid month"
```

Writing code can be a tedious and repetitive task, especially when it comes to debugging and testing. It can be especially frustrating when you're working on a large project and you're trying to track down a specific bug that's causing the program to crash. Testing code can also be boring, as it often involves running the same tests over and over again to ensure that the code is working correctly.

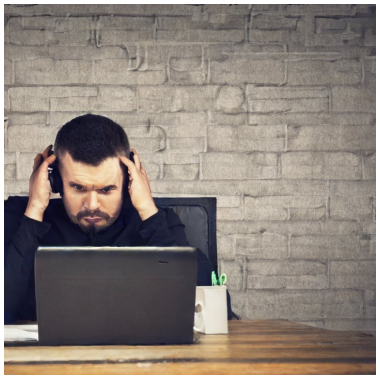


Figure 1.1: "a frustrated programmer writing boring rules on his computer" made with Stable Diffusion 2.1

Additionally, writing code can be boring because it requires a lot of concentration and focus. It can be easy to get lost in the details and lose track of time, especially if you're working on a complex problem. It can also be challenging to come up with creative solutions to problems, and it can be frustrating when your code doesn't work as expected.

While writing and testing code can be rewarding and fulfilling, it can also be a tedious and boring process. It requires a lot of patience, persistence, and attention to detail, and it can be easy to get frustrated and lose motivation. However, with practice and perseverance, it is possible to overcome these challenges and find enjoyment in the process of writing and testing code.

AI has traditionally operated by explicitly codifying human knowledge into machine-readable formats by doing the boring job of coding. This approach, which I'm calling "codified human knowledge" relies on humans to carefully structure and organize information in a way that can be understood by the AI system. The AI system then uses this structured knowledge to make decisions and perform tasks.[6]

6: Some smart people think that AI spells the "End Of Programming", and I think the headline is clickbait but the general idea that programming is changing along with advances in AI is true. We are all data scientists now!

However, recent advances in AI have largely ignored the knowledge representation problem and instead have focused on using statistical techniques and neural networks to automatically learn patterns and relationships in data. This approach, known as "deep learning," involves training large neural networks on vast amounts of data, allowing the AI system to make educated classifications and transformations of data without explicit human guidance.

Deep learning has proven to be highly effective in a variety of applications, such as image and speech recognition, and has contributed to the rapid progress we have seen in AI in recent years. However, the reliance on large amounts of data and the lack of transparency in these models can make it difficult to understand how they are making decisions, which can be a concern in certain applications (hence the title of this book).

1.4 IBM Tries Every Possible Chess Move

Deep Blue was a revolutionary computer developed by IBM that was specifically designed to play chess at the highest level. It was programmed with a vast database of chess knowledge and was able to analyze millions of positions per second.

Garry Kasparov was the reigning world chess champion at the time, and he was considered to be one of the greatest players in history. He was beaten by Deep Blue, which used rules based GFOAI (Good Old Fashioned AI) to essentially caluclate every possible move and project that move to the end of the game, then choose the best position by brute force.[7] Despite Kasparov's best efforts, he was no match for Deep Blue's computational power. In the end, Deep Blue emerged victorious, defeating Kasparov in a historic match that changed the world of chess forever.

Deep Blue was a turning point in the development of AI, but Deep Blue's methods (namely calculating every possible outcome of a Chess game to determine the best move) was not suitable for many of the world's problems. It turns out that Chess is fun, but the world is not like chess.[8] The "real"[9] future of AI was being developed elsewhere, using statistics and a toy model of the brain to solve a very practical problem for banks.

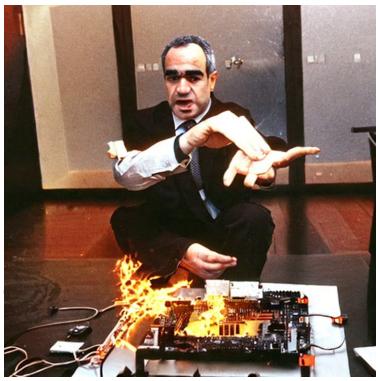


Figure 1.2: "Garry Kasparov setting a computer on fire" made with Stable Diffusion 2.1

8: if you would like to read an example of the simplest chess engine that I could imagine that is written in a smililar "codified" way as Deep Blue check out sunfish.

9: We might go back, and try again to more explicitly code everything up, and in some cases we still need to, but from the author's perspective, we live in a deep learning/neural network world.

1.5 Statistical Analysis of Handwriting is the Way of the Future

It was the early 1990s and Yann LeCun was a researcher at Bell Labs in New Jersey. At the time, the process of reading and processing checks was a tedious and time-consuming task that was done manually by bank employees. LeCun saw the potential for using artificial intelligence to automate this process, and he began experimenting with using convolutional neural networks (CNNs) to recognize patterns in images of checks.

At the time, CNNs were a relatively new type of neural network that had been developed in the 1980s for image recognition tasks. They were inspired by the structure of the human visual system, and were able to process images in a way that was similar to how the human brain does.

LeCun's work was groundbreaking, and he was able to achieve impressive results using CNNs to process checks. By 1993, he had developed a system that was able to read and process checks with a high degree of accuracy, significantly reducing the amount of time and effort that was required to process checks manually.[10]

LeCun's work on using CNNs for check processing was a major milestone in the field of artificial intelligence, and it laid the foundation for the development of many other applications of CNNs in the years that followed. Today, CNNs are widely used in a variety of applications, including facial recognition, image classification, and natural language processing. [11]

10: CNN digit OCR models are frequently featured in beginner training tutorials for deep learning libraries like PyTorch and Tensorflow, check one out on Github.

11: check out Yann LeCun domonstrating a convolutional neural network in 1993 at youtube.com.

1.6 Less Programmer Intelligence and more Data Intelligence

I think it's useful to separate the knowledge in the AI problem-space into two groups. The data and the programmer together make the programs that we use every day, and for the rest of this book I'll try and separate the discussion of the smarts of each to help us better understand the world. [12]

Figure 1.3: "a group of computer programmers striking outside of Microsoft's offices with placards saying 'rule-based programming is boring'" made with Dall-E 2

Early AI relied heavily on a human programmer to design, write, and debug computer programs. Good progammers needed domain expertise, problem-solving skills, logical thinking, and the ability to learn and adapt to new programming languages and technologies.

We are now in the age of big data, and everyone knows that "data is gold". Statistical AI methods that are now most prevalent rely on extracting meaningful insights and knowledge from large datasets. This involves using statistical and analytical methods to discover patterns and trends in data, and using this information to inform business decisions or solve problems.

Throughout this book I'll discuss the interaction between programmer and data, and what can go wrong. Working with big data and statistics at a large scale has given AI tremendous abilitiy, but has made understanding and testing models infinitely more difficult. It is your authors belief that understanding the nuance of this interaction between programmer and data is essential to understadning modern AI.

1.7 From Explicit Rules to a Black Box, and Beyond

Artificial intelligence (AI) has come a long way since its inception, and the way that it makes decisions has changed significantly over time. In the early days of AI, explicit rules were used to tell the AI system what to do in certain situations. These rules were often written by humans and encoded into the system, and the AI would follow them to make decisions.

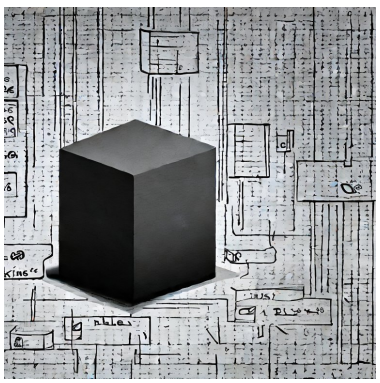


Figure 1.4: "from explicit rules, to a black box and beyond" made with Stable Diffusion 2.1

However, with the advent of deep learning, we have started to rely on a statistical understanding of the truth for AI to make decisions. Deep learning is a type of machine learning that involves training artificial neural networks on large datasets. These neural networks are able to learn patterns and relationships in the data, and can use this knowledge to make decisions.

The use of deep learning has led to the development of powerful AI that is able to perform tasks that were previously thought to be impossible for a machine to handle. For example, deep learning has led to the development of AI systems that are able to recognize faces, translate languages, and even beat humans at complex games like chess and Go.

While deep learning has led to significant advances in AI, it has also made it harder to debug and understand how the AI system is making its decisions. With explicit rules, it was relatively easy to understand why the AI made a particular decision. However, with deep learning, it is often difficult to understand exactly how the AI arrived at its decision. This can make it challenging to troubleshoot problems with the AI system and to ensure that it is making decisions that are fair and unbiased.

AI has come a long way since its early days, and the way that it makes decisions has changed significantly over time. While explicit rules were once used to tell the AI what to do, we now rely on a statistical understanding of the truth for AI to make decisions. This has led to the development of powerful AI that is able to perform a wide range of tasks, but it has also made it harder to debug and understand how the AI is making its decisions.[13]

13: From here on out I'll try and use "GOFAI" (Good Old-Fashioned Artificial Intelligence) to describe the rules-based approach, and use "Deep Learning" or "Machine Learning" to talk about modern neural network and statistics-based techniques

1.8 Key Takeaways

- **AI is a misleading term**, we should instead talk about rules-based programming (GOFAI) and deep learning so we don't confuse ourselves, our partners and our users.[14]
- Good Old-Fashioned AI **(rules-based AI) is hard to create and maintain**. We used to use it for chess engines (like Deep Blue) and translation, but now programmers favor using statistical machine learning techniques.
- Good Old-Fashioned AI is not well suited to many problems, like machine translation and handwriting detection.
- Modern Deep Learning techniques use data to train models instead of humans explicitly writing rules, and the **deep learning models are often as good as the data they are trained with**.[15]

14: Avoid the imprecise words "Algorithms" and "AI" and instead consider the more precise "rules-based AI" AKA "GOFAI" to describe situations where programmers explicitly write rules that they design and "deep learning" or "machine learning" where programmers create models using a dataset and let the machine figure out the rules via a neural network.

15: More *intelligent* data and maybe less intelligent programmers, or maybe programmers who are better trained in statistics and complex systems theory instead of "classical" computer science.

The Regression Theory of Everything

2

"AI Scientists disagree as to whether these language networks posess true knowledge or are just mimicking humans by remembering the statistics of millions of words. I don't believe any kind of deep learning network will achieve the goal of AGI [Artificial General Intelligence] if the network doesn't model the world the way the brain does. Deep learning networks work well, but not because they solved the knowledge representation problem. They work well because they avoided it completely, relying on statistics and lots of data instead. How deep learning networks work is clever, their performance impressive, and they are commercially valuable. I am only pointing out that they don't possess knowledge and, therefore, are not on the path to having the ability of a five-year-old child." Jeff Hawkins, 2022 [4]

2.1 Let's Avoid Knowledge Representation!

The knowledge representation problem in AI is the challenge of how to formally represent knowledge in a way that a computer can understand and reason about. This typically involves creating a set of symbols, rules, and structures that can be used to represent concepts, relationships, and other types of information. The goal is to create a representation that is both expressive enough to capture all relevant aspects of the domain, and computationally tractable enough to allow for efficient reasoning and inference. There are many different approaches to knowledge representation, including logic-based, semantic networks, frames, and ontologies, each with their own strengths and weaknesses.

Figure 2.1: "representing knowledge to a comupter" by Stable Diffusion 2.1

Deep learning techniques handle knowledge representation differently than traditional symbolic AI methods. Unlike symbolic AI, which relies on explicit and hand-coded representations of knowledge, deep learning techniques learn to represent knowledge implicitly through the use of neural networks.

In deep learning, knowledge is represented in the form of the weights of the neural network. These weights are learned through training on a large dataset and they capture the underlying relationships and patterns in the data. The neural network can then use these learned weights to make predictions, classifications or generate new data.

Deep learning models can handle large and complex datasets, and can automatically extract features from the data without the need for manual feature engineering. This makes them particularly well-suited for tasks such as image and speech recognition, natural language processing, and other areas where large amounts of data are available. However, they are not as good at explicating how they arrived at a decision, which can be a disadvantage.

In summary, deep learning techniques handle knowledge representation by learning the underlying patterns and relationships in the data through the use of neural networks, which can then be used for prediction, classification, and generation tasks.

2.2 A Simple Neural Network is also a Linear Regression

A neural network can be mathematically equivalent to a regression or a decision tree under certain conditions.

A neural network is a machine learning model composed of layers of interconnected artificial neurons, which are designed to process and analyze data. They can be used for a wide range of tasks, such as image and speech recognition, natural language processing, and prediction.

A regression is a statistical method used to predict a continuous variable based on one or more input features. A linear regression, for example, is a simple neural network with one input layer, one output layer and no hidden layers. In this case, the weights of the network are the coefficients of the linear equation and the network is equivalent to a linear regression model.

A decision tree is a tree-based model used for classification and prediction tasks. It consists of a series of if-then rules that are used to make decisions based on the input data. A neural network with one input layer, one output layer and one hidden layer with the ReLU activation [1] function is equivalent to a decision tree. This is because the ReLU activation function allows the network to implement a piecewise linear function which can represent the decision boundaries of a decision tree.

In summary, under certain conditions, a neural network can be mathematically equivalent to a linear regression or a decision tree. These conditions include having one input and one output layers, and having a specific activation function in the case of a decision tree.[5]

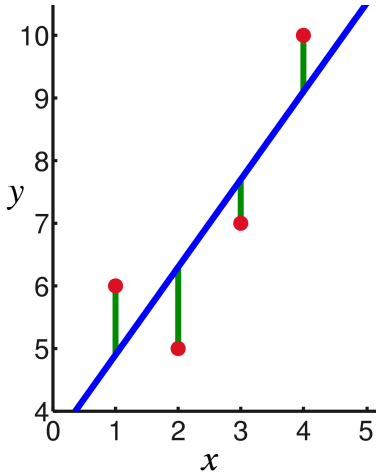


Figure 2.2: A simple linear regression, the red points are the training data, and the blue line is the regression line.

1: Neurons in a neural network can have different activation functions, if you don't know what it is and don't want to read about it in wikipedia, that's OK.

[5]: Aytekin (2022), *Neural Networks are Decision Trees*

2.3 Dummy Variables for Dummies (Wonkish)

Chapter Summary: It's all numbers, man. Machine learning techniques require that we turn everything (images, text, sound) into numbers and shove them into the model in the same way we use dummy variables in a simple regression. If you are satisfied with this, please skip this section. If you would like to learn a bit about the details and see some code examples, please keep reading. This section is necessarily technical, but should be approachable for anyone who has taken a college statistics class. [2]

Dummy variables are used in regression analysis to include categorical variables in a model. Categorical variables are variables that take on a finite number of distinct values, such as "red", "green", "blue" or "yes", "no". Since

2: Thanks to Paul Krugman for popularizing (to me at least) the *wonkish* classifier, I just mean it's a little nerdy. But it's important.

these variables cannot be directly included in a regression model, as they are not numerical, they need to be transformed into numerical variables.

The process of creating dummy variables is also known as one-hot encoding. It involves creating a new binary variable for each category of the original variable. For example, if you have a categorical variable "color" with three categories: "red", "green", "blue", you would create three binary variables: "color_red", "color_green", "color_blue". Each binary variable would take a value of 1 if the original variable is equal to the category, and 0 otherwise.

When using dummy variables in a regression, it is important to remember to include only n-1 binary variables, where n is the number of categories in the original variable. This is because including all n binary variables would result in perfect multicollinearity, which is when two or more independent variables are perfectly correlated. One of the binary variables can be dropped to avoid this problem.

Dummy variables are used in regression analysis to include categorical variables in a model. The process of creating dummy variables involves creating a new binary variable for each category of the original variable and one-hot encoding it. It is important to remember to include only n-1 binary variables, to avoid perfect multicollinearity.

The creation of dummy variables in a regression is analgous to preprocessing image, text and other data for a nueural network for deep learning. This preprocessing is important as it ensures that the data is in a format that can be easily understood and processed by the network. The preprocessing steps for numbers, text, and images are slightly different.

For numbers:

- Normalization: It is common to normalize the input data by scaling it to have a mean of 0 and a standard deviation of 1. This helps to ensure that all input features have similar scales and prevents any one feature from dominating the network's computations.
- Imputation: Handling missing data is important, as it can negatively impact the model's performance. Common imputation techniques include replacing missing values with the mean, median, or mode of the feature.

For text:

- Tokenization: Text data must first be converted into a numerical format that can be understood by the network. This is typically done by tokenizing the text into individual words or n-grams and then encoding them as integers or real-valued vectors. A one-hot encoding exactly like the dummy variable method used in regression is also frequently used. [3] [4]
- Stop-words removal: The most common words in any language like "a", "an", "the", etc. that do not contain much meaning are called stop-words, they are often removed to reduce the dimensionality of the data.

Listing 2.1: Mapping text to numbers.

```
'movies': 99,
'after': 100,
'think': 101,
'characters': 102,
'watch': 103,
'two': 104,
'films': 105,
'character': 106,
'seen': 107,
'many': 108,
'being': 109
```

3: Sometimes text is just mapped to a number! Shocking, but it works. Read more about word embeddings in the Tensorflow official tutorials.

4: GPT-3 uses byte-level Byte Pair Encoding (BPE) tokenization and has a vocabulary size of 50,257.

- Stemming/Lemmatization: Words that have the same meaning can be stemmed or lemmatized to reduce the vocabulary size and increase the chances of generalization.
- Vocabulary Size: Each model must choose a vocabulary size or the maximum number of tokens that it will analyze. This may cause misspellings, slang or typos to be discarded in analysis.

For images:

- Converting to RGB or Greyscale: Each image is analyzed by its pixel color value, every point on an image will either have 3 color values (red, green, blue) or one single value (on a white/black scale) if the image is analyzed in greyscale.
- Convolutions: Pixel values are analyzed in groups that are defined by the model, since individual pixel values are only colors (or greyness) they must be combined together by the model to detect patterns like faces and stop signs. The method of convolution is defined by the model itself.
- Resizing: neural network can only accept images of a fixed size, so resizing the image to match the network's requirements is important.
- Normalization: It is common to normalize the pixel values to be in the range of 0-1 or -1 to 1. This will help the model converge faster.
- Data Augmentation: To increase the amount of data and prevent overfitting, common data augmentation techniques such as flipping, rotation, and cropping can be applied to the images.

In summary, preprocessing is an important step in training a neural network, as it ensures that the data is in a format that can be easily understood and processed by the network. The preprocessing steps for numbers, text, and images involve normalization, imputation, tokenization, stop-words removal, stemming/lemmatization, resizing and data augmentation.

2.4 Try That Again With 33,640,010 Parameters

In the first section of this chapter we introduced the idea that a simple neural network is mathematically equivalent to a regression. This is true and a useful way to think about neural networks and deep learning. [5]

In practice, neural networks frequently trained with millions of parameters. Here is an example of the trainable parameters of a simple image classifier. This example is a dense neural network to classify handwritten digits, and is not yet as sophisticated as the one used by Yann LeCun and and company in the 1990s

In these next few examples, don't worry if you don't understand the layer types or know what batch nomalization means. The point I would like to make is that neural networks are often created with millions of trainable parameters, once you agree with me regarding this point we will discuss the implications of this in more depth.

5:

$$y = Wx + b \tag{2.1}$$

There's your simple formula for a single-cell neural network and regression, in practice we'll change W, x and b to matricies and introduce nonlinear activiation function f so,

$$y = f(Wx + b) \tag{2.2}$$

if you please. If equations scare you, don't worry about it for now.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 64)                50240
_________________________________________________________________
batch_normalization (BatchNo (None, 64)                256
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
batch_normalization_1 (Batch (None, 64)                256
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
batch_normalization_2 (Batch (None, 64)                256
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                650
=================================================================
Total params: 59,978
Trainable params: 59,594
Non-trainable params: 384
_________________________________________________________________
```

59,594 parameters! That's not too bad for a simple neural network, but in practice the networks often get even bigger. Here is a *real* example that will make your brain a little hot.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 64)        640
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 14, 14, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 14, 14, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 7, 7, 128)         512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 3, 256)         1024
```

```
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 3, 3, 512)         1180160
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 3, 3, 512)         2359808
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 3, 3, 512)         2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 512)         0
_________________________________________________________________
batch_normalization_3 (Batch (None, 2, 2, 512)         2048
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 2, 2, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 1, 1, 512)         2048
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 4096)              2101248
_________________________________________________________________
dropout (Dropout)            (None, 4096)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                40970
=================================================================
Total params: 33,642,954
Trainable params: 33,640,010
Non-trainable params: 2,944
_________________________________________________________________
```

When a neural network has millions of trainable parameters and deep layers of neurons, it can be difficult to explain to a human what each of those parameters represents or how they contribute to the overall function of the network. This is because the interactions between the different layers and neurons can be complex and non-linear, making it challenging to understand the specific role of each parameter. The parameters model complex interactions between the inputs, making it difficult to understand their specific function. Furthermore, in deep neural networks, the high number of layers can lead to high level of abstraction, meaning that the individual neurons and their weights have little interpretability.

When a neural network has millions of interacting parameters, it can lead to mathematical chaos [6], which is a phenomenon where small changes in the initial conditions of the network can lead to vastly different outputs. This is because the interactions between the large number of parameters can create non-linear relationships that are sensitive to small changes. This can

6: Mathematical chaos is a real thing! Mathematical chaos refers to the behavior of certain dynamic systems that are highly sensitive to initial conditions. This sensitivity leads to seemingly random and unpredictable behavior, even though the underlying equations governing the system are deterministic (meaning they are transparent and known to us).

make it difficult to predict the behavior of the network, as small changes in the input or the parameters can lead to unexpected and seemingly random outputs.[7]

7: Chaotic outputs in a large deterministic system are OK in many domains, and there are controls that can be put in place to keep AI "safe" but deep learning by itself will not produce those controls.

2.5 Multicolinearity and the End of Science

Data science is a horrible term because it implies that data scientists are scientists and that the work they do is scientific when in reality data scientists are not scientists and the work they do is not scientific. Data scientists use mathematics, statistics, and computer science to analyze data, but it is not scientific in the traditional sense. Data scientists do not use the scientific method and do not conduct experiments or develop theories. Data science is more akin to engineering or data analytics than actual scientific research.

The main goal of a scientist is to gain knowledge and understanding of the natural world through research, experimentation, and data analysis. Scientists make models of the world to test and explain. So-called data scientists make models too but a model with layers of interacting parameters, trained under chaotic conditions is almost impossible to explain. Multicolinearity is just one issue, but deep learning techniques are practically incompatible with the scientific method. With deep learning methods it can be difficult to determine which variable had the most impact on the outcome of the model. Coefficients of the model to be unstable and unreliable, which can make it difficult to explain the model in a meaningful way.

Explaining the inner-workings of a neural network with millions of interacting parameters is difficult for many reasons. Most of the complexity is due to the number of parameters, which can make it difficult to understand how the model works and why it makes certain decisions. The interactions between the parameters are often non-linear and can be difficult to understand, as the model can be sensitive to small changes in the parameters[8]. This can make it difficult to explain why the model is making certain decisions, as the interactions between the parameters are not always clear.

8: leading to mathematical chaos

2.6 Let's Test Some Random Inputs! Feature Importance and Explainability

With a huge complex system of neurons that combine inputs in novel ways, it becomes vary hard to understand which inputs are the most important for the system as a whole. To better understand deep learining models, data scientists use randomized or averaged inputs to a model to test the feature importance of a deep learning neural network. This type of testing is done to determine which features are most influential in the overall output of the network. This is done by randomly or averaging the input values for each feature and then running the model to see how the output is affected. For example, if a neural network is used to detect objects in an

image, the data scientist may randomize the color of the objects to see how the model's accuracy is affected. If the accuracy drops significantly, they can infer that color is an important feature in the network.

Randomized or averaged inputs to a model can also be used to determine if a particular feature is necessary for the network to function correctly. For example, if the output of the network is not as accurate when a particular feature is randomized or averaged, then the data scientist can infer that this feature is important for the network's performance. By using this method, data scientists can gain insights into which features are important and which can be removed from the model to improve performance.

2.7 The Universal Machine Learning Workflow

The Universal Machine Learning Workflow[6] is an important chapter in a technical guide for data scientists by the author of the most popular machine learning framework in the world[9] that outlines the *Universal* workflow for machine learning projects. I think this chapter should be understood by everyone using, investing in and creating machine learning models.

[6]: Chollet (2022), *Deep learning with python, second edition*

9: It's called TensorFlow and was one of the most popular and "Most Loved" Technologies of 2022

Before Chollet gets into the details of model building, he chooses to begin with a note on ethics:

"You may sometimes be offered ethically dubious projects, such as "building an AI that rates the trustworthiness of someone from a picture of their face." First of all, the validity of the project is in doubt: it isn't clear why trustworthiness would be reflected on someone's face. Second, such a task opens the door to all kinds of ethical problems. Collecting a dataset for this task would amount to recording the biases and prejudices of the people who label the pictures. The models you would train on such data would merely encode these same biases into a black-box algorithm that would give them a thin veneer of legitimacy. In a largely tech-illiterate society like ours, "the AI algorithm said this person cannot be trusted" strangely appears to carry more weight and objectivity than "John Smith said this person cannot be trusted," despite the former being a learned approximation of the latter. Your model would be laundering and operationalizing at scale the worst aspects of human judgement, with negative effects on the lives of real people.

Technology is never neutral. If your work has any impact on the world, this impact has a moral direction: technical choices are also ethical choices. Always be deliberate about the values you want your work to support."[6]

Chollet uses the outline below to explain the *Universal* workflow. I'll summarize the workflow and my own notes for a nontechnical audience here:

1. Define the Task
    a) Collect a Dataset [10]
    b) Understand Your Data [11]
    c) Choose a Measure of Success [12]
2. Develop a Model

10: This is often the hardest part. Data is often labeled by hand or stolen from another domain. See Transfer Learning (Wikipedia)

11: This is easier said than done, and requires the Machine Learning Engineer to understand the domain that they are modeling in.

12: Accuracy isn't everything! We might prefer a less accurate model that avoids false-positives in some scenarios.

a) Prepare the Data [13]
b) Choose an Evaluation Protocol
c) Beat a Baseline [14]
d) Develop a model that overfits [15]
e) Regularize and Tune Your Model

3. Deploy the Model

a) Explain Your Work to Your Stakeholders and Set Expectations [16]
b) Ship an Inference Model
c) Monitor Your Model in the Wild
d) Maintain Your Model

13: This is discussed in section 2.3 above, everything gets turned into numbers.

14: Does your model predict better than a totally random model and/or coin-flip?

15: Make your model overfit to the data! Then back off.

16: This is almost never done. And as Chollet explains this is both difficult and requires mutual understanding and domain knowledge from the part of "the business" and the Machine Learning Engineer.

Of all of the steps in the workflow, "Defining The Task" and "Explaining The Work To Shareholders and Setting Expectations" are where the most miscommunication occurs.

In defining the task, machine learning engineers are often given impossible problems to solve, and because they want to keep paying their mortgage they solve another related problem instead and allow stakeholders to jump to their own conclusions. By clearly understanding what a model is predicting and how the data is collected some of this misunderstanding can be avoided.

I will discuss Large Language Models later, but they all do the same thing. They predict the next words in a sentence, just like the keyboard on your iPhone. They come up with amazing text but by fundamentally understanding what they are predicting you gain insight into their limitations. They are also all trained on the text publicly available on the internet, this includes scientific sources, but also fan-fiction and anime, so the idea that a model trained ot this data could be relied on to predict anything truthful is preposterous.

Many models work like magic and many users assume models are predicting the future, when they are really correllating based on their past data. Even simple models of creditworthiness might have as similar problem. While building a creditworthiness model for a bank, due to lack of data a machine learning engineer might [17]create a model that predicts whether someone is a smoker instead of whether they are creditworthy. Because smoking and poverty are correllated, maybe the model "works", but it doesn't do what stakeholders think it does.

17: On purpose or inadvertently.

Setting expectations is another hellscape of misaligned incentives. Good machine learning engineers and data scientists are supposed to educate and advise stakeholders about the limitations of their own models. Read Chollet's advice on this topic below:

Success and customer trust are about consistently meeting or exceeding people's expectations. The actual system you deliver is only half of that picture; the other half is setting appropriate expectations before launch.

The expectations of non-specialists towards AI systems are often unrealistic. For example, they might expect that the system "understands" its task and is capable of exercising human-like common sense in the context of the task. To address this, you should consider showing some examples of the

failure modes of your model (for instance, show what incorrectly classified samples look like, especially those for which the misclassification seems surprising). [18] [19]

They might also expect human-level performance, especially for processes that were previously handled by people. Most machine learning models, because they are (imperfectly) trained to approximate human-generated labels, do not nearly get there. You should clearly convey model performance expectations. Avoid using abstract statements like "The model has 98 percent accuracy" (which most people mentally round up to 100 percent), and prefer talking, for instance, about false negative rates and false positive rates. You could say, "With these settings, the fraud detection model would have a 5 percent false negative rate and a 2.5 percent false positive rate. Every day, an average of 200 valid transactions would be flagged as fraudulent and sent for manual review, and an average of 14 fraudulent transactions would be missed. An average of 266 fraudulent transactions would be correctly caught." Clearly relate the model's performance metrics to business goals.

You should also make sure to discuss with stakeholders the choice of key launch parameters—for instance, the probability threshold at which a transaction should be flagged (different thresholds will produce different false negative and false positive rates). Such decisions involve trade-offs that can only be handled with a deep understanding of the business context.[6]

One does not need to be a psychologist to understand that these conversations almost never happen. The worflow of machine learning is universal, we are all doing fancy regressions and we all deal with the same wild expectations, shitty data and misalignment with business. [20]

18: Make sure you show your boss and your investors how shitty your model can be!

19: The CEO of OpenAI wrote this and still raised ten billion dollars: *"ChatGPT is incredibly limited, but good enough at some things to create a misleading impression of greatness. it's a mistake to be relying on it for anything important right now. it's a preview of progress; we have lots of work to do on robustness and truthfulness."*

20: It's worth noting that a trend in rich-world AI companies is to be a nonprofit, limited-profit or give away models for free as open-source software. My guess is that many that go this route just think that businesspeople are too stupid to understand what they are doing, so avoid doing business altogether.

2.8 Key Takeaways

- **test** exit

Decision Making, Uncertainty and Chaos

3

"I hope for some sort of peace—but I fear that machines are ahead of morals by some centuries and when morals catch up there'll be no reason for any of it." Harry Truman, 1945 [7]

3.1 Concept Drift

3.2 Theories of Creativity

Is creativity combining exsiting things, or is it coming up with something new?

Does progress slow down becasue we heavily rely (more than usual) on the work of the past to generate future work?

We always rely on the past to create future work... this is not new

The prompt you gave to ChatGPT is the creative act now. Just like you can do boring things with a paint roller, or do somethig creative.

Input and output... the AI is just a machine, a static but complicated one. Small changes in inputs will make wildly new outputs.

3.3 Getting creative with a Mikita Drill

Modern Artificial Intelligence and a Mikita drill are two complex but ultimately deterministic systems. Though they are both complex and require a user to understand how to use them, the user input ultimately decides the output.

Modern AI is a complex system of algorithms, data, and analytics that can be used to solve complex problems. AI systems can learn from data, identify patterns, and make predictions about the future. AI systems are typically used to automate and assist human decision making. AI systems are programmed with specific objectives and goals, and the user input decides the output. For example, an AI system could be programmed to solve a mathematical problem and the user input would determine the parameters of the problem and the output would be the solution.

A Mikita drill is also a complex deterministic system. The user input is limited to the type of drill bit, the speed of the drill, and the direction of the drill. The output is determined by these inputs, as the drill will only drill in the direction and speed determined by the user. The user also has to choose the correct drill bit to ensure the drill can do the job correctly and safely.

Overall, both modern AI and a Mikita drill are complex but ultimately deterministic systems. The user input, however limited it may be, ultimately controls the output of the system. Both systems require a user to understand how to use them and to make the correct input to get the desired output.

Creative ways to use a drill.

- What you point it at
- The bits you use
- The pace and speed of your finger on the trigger
- The amount of pressure you put on the drill and the wall

3.4 From Spaghetti Code to Lasagna Neuron Layers

These layers are totally transparent, but you can't understand them because they're complicated :)

3.5 Garbage In, Garbage Out

You are essentially programming with data, so if your data sucks so will your prediction, you also really can't generalize, only correllate.

3.6 Garbage In, New Perspective Out?

What about cross-domain models, where maybe I train with a poetry dataset, and point my language model on nonfiction.... hmmm.

3.7 The Impossibility of Fairness

Representation, "fixing the training set" [8], or the Impossibility of Fairness from a model.

[8]: Christian (2020), *The Alignment Problem: Machine Learning and Human Values*

3.8 What Are We Prediciting Again?

are you predicting the right thing? Are you really predicting how valuable the company is or just whether it'll be the next meme stock?

3.9 The Perverse Incentives of Data Scientists: Job Security by Obscurity

Investors want predictions, Data Scientists don't have good data, but want to keep their jobs. Bad science rolls downhill to the user, and takes a long time to snuff out.

3.10 Humans Love Computers

Working together seems like a good idea, but how. Talk about 2 percent model control and model uptake.

Also talk about Kasparov's tournament and Thinking for yourself in the age of AI. [**mansharamani_2020**]

mansharamani_2020

Classification and Profiling: Studies in AdTech and Policing

4

4.1 Everything is Clasification

4.2 Classification in Reverse; Generating the Stereotype

4.3 Code Your Own Classifier in 5 Minutes!

Beginners tutorial

4.4 The End of Nuance

The unclassified?

4.5 If You Were Unable To See A Doctor, is it OK?

Increasing access of skin cancer detection.

4.6 Lazy Thinking

Online Advertising, Justice, Job Applications, Creditworthiness, Getting Insurance (Weapons of Math Destruction), Civic Life, /sideciteOneil2017 ; The Default Male, Invisible Women effects snow clearing schedules and drug discovery

4.7 Data Where There Is None

Talk about how generative tools can create data to be classified (like LAION).

Self-Driving With Statistics | 5

5.1 Trolley Problems

5.2 The Third Rail

5.3 Purple Lights and Changes In Fashion

5.4 Multicolinearity (Reprise)

5.5 See You In Court, Asshole!

5.6 A Train is a Self-Driving Car, Right?

Discuss how the problem space has changed in warehousing, we don't actually have self-driving forklifts, we have moving shelves.

Maybe talk about elon musk and manufacturing.

Generating Art: Avante-Garde or Derivative?

6

"(Traditional) search engines are databases, organized collections of data that can be stored, updated, and retrieved at will. (Traditional) search engines are indexes. a form of database, that connect things like keywords to URLs; they can be swiftly updated, incrementally, bit by bit (as when you update a phone number in the database that holds your contacts).

Large language models do something very different: they are not databases; they are text predictors, turbocharged versions of autocomplete. Fundamentally, what they learn are relationships between bits of text, like words, phrases, even whole sentences. And they use those relationships to predict other bits of text. And then they do something almost magical: they paraphrase those bits of texts, almost like a thesaurus but much much better. But as they do so, as they glom stuff together, something often gets lost in translation: which bits of text do and do not truly belong together." Gary Marcus, 2023 [**marcus_2023**]

6.1 Predictive Keyboards

6.2 GPT On A Napkin

Discuss this [1]

1: GPT On A Napkin

6.3 Compress The World's Achievements Into One Database

Lossy lookup of data, see how close you get to creating famous artworks or wikipedia pages?

Talk about AI upscaling and AI as lossy compression. we stored the entire corpus of human knowledge, but AI retrieves it in a silly way.

6.4 Upscaling What Is Lost

How to leverage this stuff.

6.5 Who Owns This Stuff Anyway?

Discuss FSF and Copilot, lawsuits.

6.6 Use Critical Thinking to Become A Critic

Talk about ChatGPT, deterministic vs probabalistic and Thomas Hobbes
[2]

2: AI Homework

Revolutionary for Whom? 7

"The inhabitant of London could order by telephone, sipping his morning tea in bed, the various products of the whole earth – he could at the same time and by the same means adventure his wealth in the natural resources and new enterprise of any quarter of the world – he could secure forthwith, if he wished, cheap and comfortable means of transit to any country or climate without passport or other formality." John Maynard Keynes, 1920 [9]

7.1 Self-Driving Horses

7.2 The Battle of the Assistants

Butler vs Indian Virtual Assistant vs Siri

7.3 Employees That Are Better Than You

Respond directly to Jon Krohn's TED talk about monkeys being dumber than us... what about construction equipmenth that's stronger than us physically, or racism/eugenics people that are dumber than us [10]

[10]: (2022), *Jon Krohn*

7.4 Who is Helped the Most?

7.5 Who is Hurt the Most?

7.6 How to Respond

7.7 This Book is a Case Study

Unplugging Skynet 8

"The second requirement of goal-misalignment risk is that an intelligent machine can commandeer the Earth's resources to pursue its goals, or in other ways prevent us from stopping it... We have similar concerns with humans. This is why no single person or entity can control the entire internet and why we require multiple people to launch a nuclear missile. Intelligent machines will not develop misaligned goals unless we go to great lengths to endow them with that ability. Even if they did, no machine can commandeer the world's resources unless we let it. We don't let a single human, or even a small number of humans, control the world's resources. We need to be similarly careful with machines." Jeff Hawkins, 2022 **[hawkins_2022]**

8.1 AI and Human Safety

8.2 Useful Incompatability

It's a feature not a bug that no single comupter can control all others, AKA Security by Obscurity

8.3 Checks and Balances

8.4 Free-Rider Problems

I can steal your AI quite easily from outputs

8.5 When You Can't Tell The Difference

Talk about Taleb's aphorisms "Another definition of modernity: conversations can be more and more completely reconstructed with clips from other conversations taking place at the same time on the planet.", "You are alive in inverse proportion to the density of cliches in your writing."

8.6 Sucker Traps

"The sucker's trap is when you focus on what you know and what others don't know, rather than the reverse."

8.7 Dead Inside

"If you know, in the morning, what your day looks like with any precision you are a little bit dead - the more precision the more dead you are."

9 Errors and Omissions

Bibliography

Here are the references in citation order.

[1] John von Neumann and Ray Kurzweil. *The Computer and the Brain (The Silliman Memorial Lectures Series)*. New Haven, CT, USA: Yale University Press, Aug. 2012 (cited on page ii).

[2] Matt Welsh. *The end of programming*. Jan. 2023. URL: https://cacm.acm.org/magazines/2023/1/267976-the-end-of-programming/fulltext (cited on page 1).

[3] Elaine Rich, Kevin Knight, and Shivashankar B. Nair. *Artificial Intelligence*. Tata McGraw-Hill, 2009 (cited on page 1).

[4] Jeff Hawkins. *A thousand brains: A new theory of intelligence*. Basic Books, 2022 (cited on page 7).

[5] Caglar Aytekin. 'Neural Networks are Decision Trees'. In: (2022). DOI: 10.48550/ARXIV.2210.05189 (cited on page 8).

[6] Franc ois Chollet. *Deep learning with python, second edition*. Manning Publications, 2022 (cited on pages 14, 16).

[7] David McCullough. *Truman*. Riverside, NJ, USA: Simon & Schuster, June 1992 (cited on page 17).

[8] Brian Christian. *The Alignment Problem: Machine Learning and Human Values*. New York, NY, USA: W. W. Norton & Company, Oct. 2020 (cited on page 18).

[9] John Maynard Keynes, Elizabeth Johnson, and Donald Moggridge. *The Collected Writings of John Maynard Keynes (Volume 5)*. Cambridge, England, UK: Cambridge University Press, Dec. 2012 (cited on page 24).

[10] *Jon Krohn*. [Online; accessed 18. Oct. 2022]. Oct. 2022. URL: https://www.jonkrohn.com/posts/2022/10/7/tedx-talk-how-neuroscience-inspires-ai-breakthroughs-that-will-change-the-world (cited on page 24).