FORGET ME NOT: MEMORISATION IN GENERATIVE SEQUENCE MODELS TRAINED ON OPEN SOURCE LICENSED CODE[*]

Ivo Emanuilov[†], Thomas Margoni[‡]

ABSTRACT

Generative sequence models, like GPT-3/4, Stable Diffusion and DALL·E, are increasingly utilised to produce artifacts traditionally associated with human ingenuity, such as text, images, audio, videos and code. Despite their impressive ability to generalise on unseen data, these models are prone to memorising fragments of their training data. In some extreme cases, these 'memories' may contain verbatim and potentially infringing reproductions of works protected by copyright. In this paper, we focus on one specific example, namely program source code.

The ongoing litigation against Microsoft's GitHub Copilot service shows that these concerns are far from theoretical. GitHub Copilot is a commercial service designed to support software development workflows. It generates code based on a program specification provided by a programmer in a natural language. The service relies on the generative model Codex, which has been trained on public open-source code repositories hosted on GitHub and fine-tuned for code generation. In the words of its creators, this model has been trained on 'billions of lines of public code', that is, computer programs arguably covered by copyright law and distributed under an open source licence.

Copilot has been shown capable of reproducing, on occasion, verbatim fragments of what is allegedly its training dataset without appropriate attribution or notice. These reproductions have included not only functional code, but also original, expressive code plausibly protected by copyright. While open source software is, by definition, distributed with its source, many open source licences follow a direct licensing model where attribution, notice and licence notice preservation requirements must be observed to avoid downstream recipients being found in breach of the licence.

This controversy has sparked heated debates in both deep learning and legal communities as to the legality of developing and using such models under copyright law. In this paper, we explore the implications of memorisation for copyright infringement under EU law and propose a set of solutions that may help alleviate these concerns.

Keywords: generative AI, memorisation, copyright, open source, text and data mining, artificial intelligence, machine learning

[*] DOI: 10.5281/zenodo.10635479

[†] Researcher, KU Leuven Centre for IT & IP Law – imec, 6 Sint-Michielsstraat, bus 3443, 3000 Leuven, email: ivo.emanuilov@kuleuven.be

[‡] Research Professor of Intellectual Property Law, KU Leuven Centre for IT & IP Law – imec, 6 Sint-Michielsstraat, bus 3443, 3000 Leuven, email: thomas.margoni@kuleuven.be

1. INTRODUCTION

In recent years, we have seen significant progress in various applications for machine translation, text autocorrection and autocompletion, speech recognition and synthesis of digital images, audio, video and code. These developments are the result of advances in machine learning, particularly the generative pre-trained transformers (GPT) architecture, which made it possible to build generative models on a scale.[1]

Transformers are a type of a deep learning architecture used train models that processes sequential input data. They typically entail self-supervised learning which involves unsupervised pretraining and supervised fine-tuning. Transformers are usually pretrained on large unlabelled datasets because of the widespread availability of unlabelled data, compared to labelled data. They are then fine-tuned using supervised learning with more limited labelled data. Transformer-based models are characterised by self-attention, which, simply put, mimics cognitive attention, i.e., the ability to focus on the more important parts of the input data at the expense of others.[2] Transformers can be used for a variety of tasks, such as computer vision and natural language processing applications, e.g., language modelling, sentiment analysis etc. Notable examples include GPT-3 and 4, DALL·E 2 and Stable Diffusion.

Transformers have proven particularly useful for generating synthetic content. This category of models, colloquially called generative models, is increasingly utilised to produce artifacts traditionally associated with human ingenuity, such as text, images, audio, videos and code. Recent examples of applications include ChatGPT, Google Bard, Microsoft Bing Chat, GitHub Copilot, and Microsoft Designer. Despite their impressive ability to generalise, that is, to correctly perform their function on new and unseen data, generative models are prone to memorising fragments of their training data. In extreme cases, these 'memories' may contain verbatim copies of works protected by copyright which, when reproduced including in downstream applications, may potentially expose the model's developers and users to risk of infringement. The recent controversy around Microsoft's GitHub Copilot or Stability AI's service has disproved the myth that these concerns are purely theoretical.

GitHub Copilot was launched as a subscription-based commercial service to support pair programming software. GitHub Copilot relies on the underlying Codex model, initially developed by OpenAI, to suggest code and entire functions in real-time directly from a code editor.[3] The service is available as a plugin for the popular code editor Visual Studio Code. It generates source code based on a program specification provided by a programmer in natural language. For example, a programmer could simply ask for a Python implementation of a matrix multiplication algorithm and the plugin would generate the corresponding source code, similarly to what ChatGPT does for natural language.

The service was built on top of the generative model Codex, which has been trained on public free and open-source software repositories and fine-tuned for code generation. In the words of its creators, this model has been '[t]rained on billions of lines of public code', that is, computer programs distributed under an open source licence. Soon after the initial launch, doubts began to surface about whether Copilot might be doing a little more than simply generalising exceedingly well on unseen data.[4] Developer reports about Copilot reproducing open source code without a corresponding copyright notice or licence text raised concerns if the model may have arbitrarily memorised, i.e., made verbatim copies of some of its training data.

This controversy has sparked heated debates over the compatibility of developing and using generative models in both deep learning and legal communities. These discussions have recently yielded the first class-action lawsuit in the US Federal Court of San Francisco against Microsoft and GitHub on behalf of a proposed class of 'possibly millions of

[1] These models are also referred to as large language models. See Daniel Jurafsky and James H Martin, *Speech and Language Processing* (Third Edition draft, 2022) 33.
[2] Self-attention can capture the dependencies between different parts of a sequence and differentially weights their significance.
[3] GitHub, 'GitHub Copilot - Your AI Pair Programmer - Frequently Asked Questions' (*GitHub*, 2023) <https://github.com/features/copilot> accessed 19 December 2022.
[4] Typically, generalisation refers to the ability of a model to perform accurately on new data pulled from the same distribution as the one used to build the model.

GitHub users'.[5] The lawsuit is built on the contention that the defendants have violated the rights of developers who have uploaded code and other works under an open source licence on GitHub. Interestingly, while the complaint provides a wealth of examples, it does not really discuss or anticipate technical aspects of the learning process which may have a bearing on any potential finding of infringement. Recently, stakeholders in other domains have taken similar spur-of-the-moment decisions to launch litigation against various generative AI players. These cases are now collectively referred to as the "Trial of AI".[6]

In this paper we concentrate on the techno-legal aspects of generative AI, focusing not only on the effect of literal reproduction (the 'what') but also on the underlying reasons for this phenomenon (the 'why/how'). We delve into the role of memorisation in large language models (LLMs) and its implications for copyright infringement under EU law. Specifically, we investigate generative statistical models created over big code extracted from publicly available open source repositories and reflect on whether a proper regulatory solution should account for the differences between generalisation and memorisation for copyright purposes. Our key message is that copyright infringement analysis should be better informed by a sound understanding of the technical processes involved in language modelling.

The paper is structured in five main parts. Following this introduction, in the second part we turn to an analysis of the architecture of one well-known generative model, namely GPT-3. The point is to provide a sound understanding of how code is embedded as data. We briefly explore the steps of encoding, embedding, positional encoding, attention, feed forward, normalisation and decoding through the lens of copyright restricted acts. In the third part, we formally define memorisation from a technical and mathematical point of view. This is followed, in the fourth part, by a copyright analysis of the process of generating code from a fine-tuned model where data becomes code. Finally, in the fifth part we advance a set of proposals on how memorisation in large language models can be addressed from a legal, technical, and organisational point of view. This part is followed by a short section with conclusions.

2. GENERATIVE LANGUAGE MODELS FOR CODE

In the current international debate on generative AI and copyright, one of key question is whether (and which of) the acts involved in the training of a generative model need a statutory or contractual permission. No authorisation means that the acts involved in AI model training are not regulated by copyright. However, if an authorisation is necessary, copyright acquires a significant regulatory function vis-à-vis AI and, more broadly, technological innovation. To find out which of these two distinct scenarios is most likely – and desirable – it is necessary to have a deeper look into the architecture of a typical generative model, such as GPT-3.

2.1. LARGE LANGUAGE MODELS (LLMS)

Large language models (LLMs) are models that assign probabilities to sequences of words given some preceding context.[7] These models are based on architectures designed to process sequential input data, such as recurrent neural networks (eg, long short-term memory) or, more recently, transformers. These models are particularly well suited for speech recognition (e.g., the sequence 'Do you know what I mean?' is much more probable than 'Do you know I am mean?'), spelling correction (e.g., 'their' and 'they're'), grammatical error correction, machine translation tasks and, as of lately, generative tasks, such as the creation of digital artifacts like images, audio, video and program source code.

By assigning conditional probabilities to every possible next word in the sequence, language models provide a distribution over an entire vocabulary.[8] For example, if our preceding context is 'I like to play the' and we want to know the likelihood of the next word in the sequence being 'guitar', then, in mathematical terms, we would need to compute the following probability:

[5] Matthew Butterick and others, 'GitHub Copilot Litigation' (*GitHub Copilot litigation*, 3 November 2022) <https://githubcopilotlitigation.com/> accessed 13 March 2023.
[6] Pendant, 'Master List of Lawsuits v. AI, ChatGPT, OpenAI, Microsoft, Meta, Midjourney & Other AI Cos.' (*Chat GPT Is Eating the World*, 19 October 2023) <https://chatgptiseatingtheworld.com/2023/10/19/master-list-of-lawsuits-v-ai-chatgpt-openai-microsoft-meta-midjourney-other-ai-cos/> accessed 26 October 2023.
[7] Jurafsky and Martin (n 1) 31.
[8] ibid 185.

$$P(guitar|I\ like\ to\ play\ the)$$

The objective of a language model is therefore to generate text that reads like natural language. However, this likeness is achieved by an algorithm that employs an approach based on statistical probability and not logical reasoning. Probabilistic measures are used to evaluate what role each word plays in its surroundings (locally) and in longer sequences (globally).[9] The self-attention mechanism of transformers, for example, gives the context for any position in the input sequence.

In recent years one language model architecture has garnered the attention of legal experts, namely generative pretrained transformers (GPT). GPT-3, which is the foundation model on which Codex has been fine-tuned, is an example of such a model. To understand how it works, we need to take a functional view of some key elements of its architecture.[10]

2.1.1. ENCODERS AND DECODERS

At a very high level, the transformer architecture consists of encoding and decoding components and the different interactions between them.

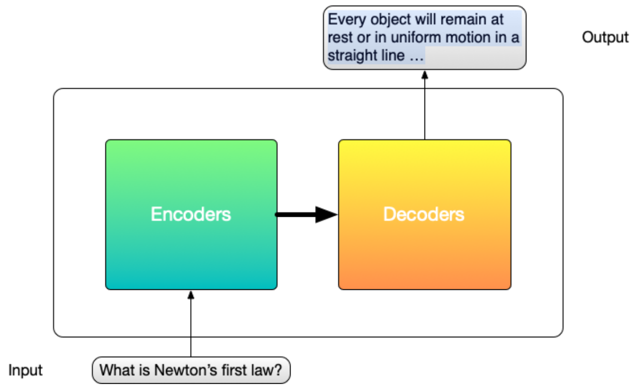
Every object will remain at rest or in uniform motion in a straight line …
Output
Encoders
Decoders
Input
What is Newton's first law?

FIGURE 1 HIGH LEVEL VIEW OF THE TRANSFORMER ARCHITECTURE

In GPT-3, there are exactly 96 encoding and 96 decoding layers with an almost identical structure. The structure of an encoder includes a feed forward neural network and a self-attention mechanism whereas decoders include an extra encoder-decoder attention mechanism.

An input sequence first goes through the encoder's self-attention layer, which gives the encoder a 'hint' on which input tokens to focus. The output of this self-attention layer is fed into a feed forward neural network which is applied to each position independently.

[9] Hannah Brown and others, 'What Does It Mean for a Language Model to Preserve Privacy?' (arXiv, 14 February 2022) 3 <http://arxiv.org/abs/2202.05520> accessed 8 November 2022.

[10] For the sake of simplicity, the exposition in the following paragraphs is not a complete and precise description of the transformer architecture. The focus is instead on the technical elements that may prove critical to the copyright analysis in the subsequent sections.

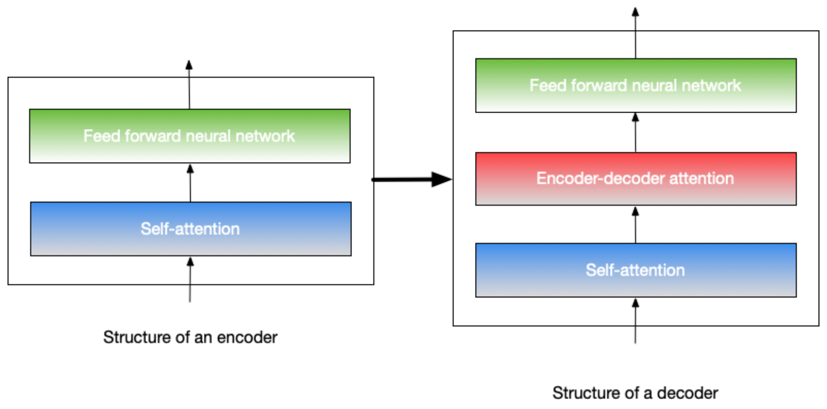
Feed forward neural network
Self-attention
Structure of an encoder
Feed forward neural network
Encoder-decoder attention
Self-attention
Structure of a decoder

FIGURE 2 STRUCTURE OF AN ENCODER AND DECODER

In simple terms, the input is a sequence of tokens (e.g., words or parts of words) and the output is a sequence of predictions of words that are most likely to occur next. The model has an autoregressive objective that maximises the probability of the next word in a sequence based on the preceding words.[11] However, the model has no *understanding* of context or words. To overcome this challenge, its algorithm uses vectors of numbers to compute the probabilities. This means that words must first be represented as numbers and the relationship between words in a sentence must be encoded computationally.

2.1.2. Word embeddings

Words are transformed into vectors by creating a vocabulary of all words in a given natural language and assigning a value to every word. In transformers, the input embedding is performed by an embedding algorithm in the very first encoder which results in so-called "word embeddings".

An embedding is a numerical representation of some kind of information, like an image or text. Its purpose is to capture the semantic meaning of the embedded object. Mathematically speaking, embeddings are dense vectors of floating-point values. These values are not specified manually but are parameters that can be figured out by an algorithm, that is to say, weights that are learned by a model. The higher the dimension of the embeddings, the better the chance it can capture fine-grained relationships between the objects (e.g., words). Importantly, embeddings carry semantic meaning about the embedded object; they are a proxy to encode the semantics.[12] However, embeddings are not a copy of that object but merely a numerical representation of it. Embeddings are derived from the parameters or weights of the model, and they encode and decode the input and output texts.

Let's illustrate this with an example. GPT-3 has a vocabulary size of 50,257 words. Typically, single words are converted into a vector of N dimensions using a technique called one-hot encoding, where N is equal to the size of the vocabulary and each position represents a word.

Let's say we have two documents in our corpus with the following content:

Document 1	'I love to play the piano'
Document 2	'My mom used to play the violin'

There are 10 unique words in the corpus, so the corpus can be represented as an index 0-9. This represents a label and an assigned integer value, such as:

$$\{'I': 0, 'love': 1, 'to': 2\ \ \ldots\}$$

[11] Autoregressive objectives are common in language modelling as they are concerned with predicting future values based on past values.
[12] Brown and others (n 9) 3.

Document 1 can now be represented as:

[0,1,2,3,4,5]

and Document 2 can be represented as:

[6,7,8,2,3,4,9]

Each of these documents can then be represented as a one-hot vector of dimension 10, i.e., the dictionary's size. Here's an example for Document 1:

[[1,0,0,0,0,0,0,0,0,0], [0,1,0,0,0,0,0,0,0,0] ...]

Text vectorisation can also be illustrated as a lookup table. To represent a word, we create a zero vector whose length is equal to that of the vocabulary (i.e., the unique words) and place a '1' in the index corresponding to the word. Here's an example of one-hot encoding of each word in the sentence "The quick brown fox jumps over the lazy dog":

TABLE 1. EXAMPLE OF ONE-HOT ENCODING

	Quick	Brown	Fox	Jumps	Over	Lazy	Dog	The
The	0	0	0	0	0	0	0	1
quick	1	0	0	0	0	0	0	0
brown	0	1	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…

The one-hot encoding of the sentence would simply be the concatenation of the one-hot vectors for each word. In the GPT family of models, this encoding is performed for every word in the sequence which results in a $2{,}048 \times 50{,}257$ matrix of just 0s and 1s. For the sake of precision, the actual method used by GPT-3 is byte-pair tokenisation (BPE), which is a data compression algorithm that encodes groups of characters instead of full words. As a vector of length $50{,}257$ is too big to compute, with most of it containing 0s anyway, an embedding function is learned to project the information about a word's meaning to a space with less dimensions.[13]

Embedding functions are a key feature of language models. They map from words or phrases to vectors in a high-dimensional space. This is done in such a way that the proximity between two vectors is indicative of the closeness of the meaning of the respective words or phrases, i.e., this allows us to computationally encode semantics.[14] In the case of GPT-3, the embedding function is performed by a neural network that takes the input vector of length $50{,}257$ and outputs an n-length vector of numbers. For example, there are $12{,}288$ embedding dimensions in GPT-3 where each dimension represents some made-up property of a word, such as length, likelihood etc.[15] Assigning a value to each dimension allows the model to know, from a statistical point of view, the exact word that we mean. Every one-hot vector is then multiplied by the learned weights which results in a large $2{,}048 \times 12{,}288$ sequence embeddings matrix.

Each encoder receives a list of vectors as input and this list is processed by passing the vectors through the self-attention layer and then through the feed forward neural network. The output of each previous layer is passed as input to the next layer. Importantly, each embedding follows its own path in the encoder, which allows for a high level of parallelisation.

2.1.3. Self-attention

The sparse attention mechanism is one of key innovations of the GPT architecture as it allows the model to predict on which input tokens it should focus. As explained, attention is a deep learning technique whereby neural networks can

[13] Daniel Dugas, 'The GPT-3 Architecture, on a Napkin' (*Robotics & Machine Learning*, 2022) <https://dugas.ch/artificial_curiosity/GPT_architecture.html> accessed 10 May 2023. Also Ashish Vaswani and others, 'Attention Is All You Need' (arXiv, 5 December 2017) 5 <http://arxiv.org/abs/1706.03762> accessed 15 November 2022.
[14] Brown and others (n 9) 3.
[15] Dugas (n 13).

learn which parts of the input data they should prioritise. Simply put, the attention mechanism looks at arbitrary data points, which are statistical representations of words, in order to compute what is more “important” depending on the given context.

At a very high level, attention allows the model to associate related words. Let’s take the following two sentences:

“Large hole appears in High Street. City authorities are looking into it.”

Despite the ambiguity of the second sentence, most people would easily read through the headline’s brevity. How would the model know, though, which word to associate with the third-person pronoun in the second sentence, if any (i.e., what are the City authorities looking into – the large hole, or the issue that it represents)? In the case of the GPT family of models, this is performed by a mechanism called multi-headed attention which involves matrix multiplication and scoring.

Recently, scientists have attempted to end-to-end reverse engineer the attention mechanism in GPT-2 using the following simple linguistic task:[16]

Given the phrase: “When Mary and John went to the store, John gave a drink to”, what would be the next word?

Expectedly, the model predicted “Mary” as the next word in the sequence. The researchers managed to isolate a circuit that was responsible for the flow of information connecting “Mary” with the next token prediction:

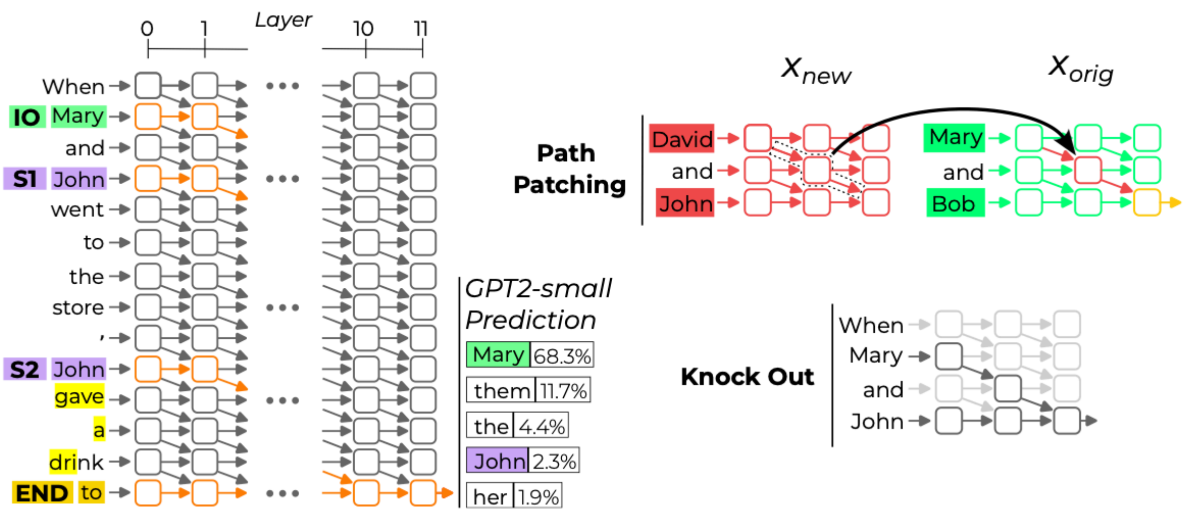

FIGURE 3 FLOW OF INFORMATION IN GPT-2 FOR A SAMPLE SEQUENCE[17]

The researchers discovered that several categories of attention heads contributed to the correct prediction. They called three of the attention heads “**Name Mover Heads**” because they attend to the correct name and copy whatever they attend to. These heads copied the information from the IO token (“Mary”) into the final output. But how did the model *know* which word to copy? The team found that four other attention heads called “**Subject Inhibition (S-Inhibition) Heads**” that are active at the END token in the figure above. These **S-Inhibition Heads** remove duplicate tokens from the Name Mover Heads’ attention and inhibit the **Name Mover Heads**’ attention to $S1$ and $S2$ tokens.

The **S-Inhibition Heads** inhibited the **Name Mover Heads**’ attention to $S1$ and $S2$ because of two attention layers the researchers called **Duplicate Token Heads**. The **Duplicate Token Heads** identify the tokens that have already

[16] Kevin Wang and others, ‘Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 Small’ (arXiv, 1 November 2022) 2 <http://arxiv.org/abs/2211.00593> accessed 7 February 2024.
[17] Reproduced from ibid.

appeared in the sentence. These heads are active at the $S2$ token, attend primarily to the $S1$ token, and signal that token duplication has occurred by writing the position of the duplicate token.[18]

Finally, the **Name Mover Heads** are active at the END token, and they copy the names they attend to. Since the S-Inhibition head 'blocks' the $S1$ and $S2$ tokens, the **Name Mover Heads** can only attend to the IO token ("Mary") and therefore the output is "Mary".[19]

2.1.4. Positional encoding

In principle, the model has no information about the position of the current token (i.e., word in our example) in the sequence. In the case of GPT-3, positional encoding is used to represent the token's current position (i.e., $[0 - 2047]$) by injecting a vector to each input embedding which provides information about the relative or absolute position of the tokens in the sequence.[20]

The position of a token is passed through $12{,}288$ sinusoidal functions with different frequencies. Sinusoidal functions allow for representing cycles of various lengths which may occur in different types of natural language, for example, legal texts, poetry, etc. This results in a $12{,}288$-dimensional vector of numbers for each word in the vocabulary. These vectors are then combined into a matrix with $2{,}048$ rows where each row represents the $12{,}288$-column positional encoding of a word (token) in the sequence. This matrix is then added to the sequence embeddings matrix. After passing through all 96 layers, the input is computed into a $2{,}048 \times 12{,}288$ matrix. This matrix contains a $12{,}288$-vector of information about which word should appear next for each of the $2{,}048$ output positions in the sequence.

In essence, the added value of the model lies precisely in this encoding. It is now only a matter of decoding to extract this information. The encoded information is extracted by reversing the mapping of a one-hot encoding of a word to a $12{,}288$-vector embedding. This transforms the output embedding into a $50{,}257$-word encoding which can be converted into a probability distribution using a normalised exponential function. In other words, the encoded embeddings contain the information needed by the decoder to produce the output.

The output of the stack of decoders is a vector of floating-point numbers. This output is transformed into a word by a linear layer and a $Softmax$ layer. The linear layer is a fully connected neural network that projects the decoder stack output onto a larger vector of the size of the vocabulary (logit vector), with each cell corresponding to the score of a unique word. The $Softmax$ layer then converts these scores into probabilities and then an $argmax$ function gets the index of the cell with the highest probability which produces the word associated with this index.

2.1.5. Transformers in a nutshell

To summarise, transformers are a model architecture for generating an output sequence from an input sequence using an encoder neural network and a decoder neural network that are both attention-based. The output sequence similarly contains a sequence of $2{,}048$ guesses, which are the probability for each expected word. This model contains probabilities but, importantly, it does not contain any actual natural language in human-readable form. The syntax, semantics and pragmatics are absent in any humanly cognisable sense of these terms. There are only words, or better bits of words, such as syllables and the numbers representing their statistical distribution given a certain input.

The full architecture is represented in the following diagram reproduced from the original transformer model architecture:

[18] ibid 4.
[19] ibid 4–5.
[20] Vaswani and others (n 13) 5–6.

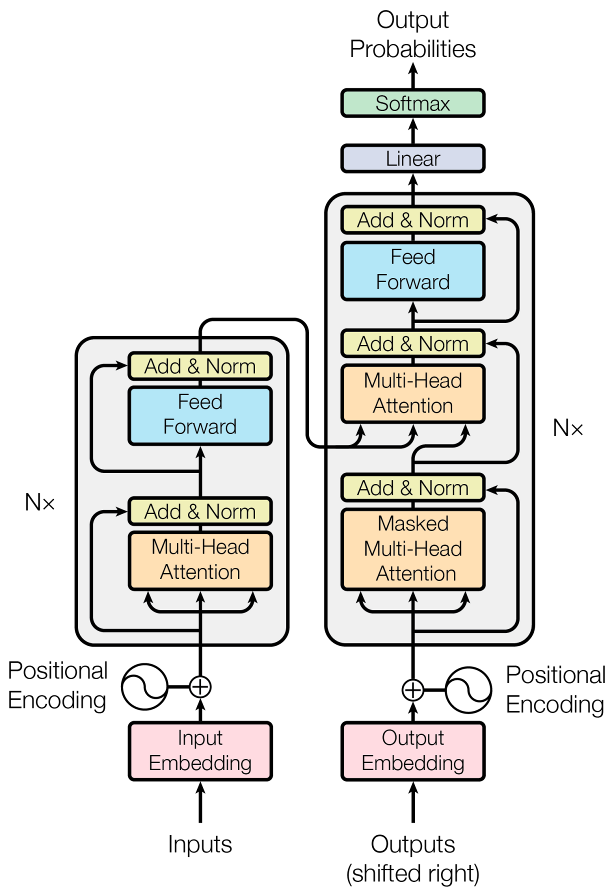

FIGURE 4. THE COMPLETE TRANSFORMER MODEL ARCHITECTURE[21]

The architecture of GPT-3 lies at the heart of Codex, which is essentially a version of GPT-3 fine-tuned[22] on ‘publicly available code from GitHub’.[23] Codex was fine-tuned on GPT-3 because the strong natural language representations in GPT-3 allowed for evaluation of natural language prompts (i.e., docstrings)[24]. The code lexer[25] in Codex is based on the GPT-3 text tokeniser[26] but it has been improved to account for the different distribution of words in computer code (as opposed to natural language) and the incorrect encoding of whitespaces.[27] Codex is evaluated using a metric

[21] Reproduced from ibid 3.
[22] Fine-tuning is a transfer learning task where a task-agnostic unsupervised language model trained on a large corpus of data is adapted for downstream tasks. This is done by performing new rounds of training to the parameters of the pre-trained models. In our example, GPT-3 is the pre-trained model and Codex is the fine-tuned model.
[23] Mark Chen and others, ‘Evaluating Large Language Models Trained on Code’ [2021] arXiv:2107.03374 [cs] 4 <http://arxiv.org/abs/2107.03374> accessed 25 November 2021.
[24] Docstrings are string literals specified in source code that are used to document or annotate code segments. Unlike typical source code comments, however, docstrings are not removed from the source tree when parsed. They are retained during runtime which allows a programmer to inspect them while the program is running. Not all high-level programming languages support docstrings, but many do, such as Python, Julia or Haskell.
[25] In compiler theory, a lexer performs a lexical analysis of lexical tokens of code, i.e., when sequences of characters are converted into strings with a meaning. Together with a parser, the lexer is used to analyze the syntax of a programming language. They are both essential to compilation.
[26] Much like a code lexer, the tokenizer classifies sections of a string of input characters, i.e., by splitting the input string into tokens, such as individual words, punctuation marks, spaces etc.
[27] Chen and others (n 23) 4.

called pass@k which tests the generated sample for functional correctness through a set of unit tests.[28] If a sample passes the tests, it is considered functionally correct. Testing for functional correctness is preferred to match-based metrics because of the large space of functionally equivalent programs that need to be compared against a reference solution.[29]

2.2. Interim conclusions

We can summarise the technical relevant aspects and their relevance from a copyright point of view as follows:

- Language models encode words as groups of character tokens numerically represented as vectors (embeddings).
- Language models use neural networks to maximise the probability of every possible next word in a sequence, resulting in a distribution over a vocabulary which consists of all words. Each input token is mapped to a probability distribution over the output tokens, that is, the following tokens.
- Language models rely on multi-headed sparse attention mechanisms to create context-aware representations of words, including their meaning and relationship to each other.
- Language models do not memorise the syntax, semantics, or pragmatics of the language in the specific pieces comprising the training data (e.g., a specific poem, short story, or code implementation). They instead learn patterns and derive rules to generate syntactically, semantically, and pragmatically coherent text. Even if the 'source code' of a large language model could be made available,[30] it would be virtually impossible to revert to the training sources (e.g., literary works such as articles, books, etc.).
- Language models are trained to minimise a loss function, that is, the difference between actual output and predicted output, over each training example in the training dataset.[31] One way to reduce a model's loss is to memorise the next token in a sequence for every prefix in the training data. Admittedly, some form of memorisation is almost unavoidable, e.g., a model needs to remember the correct spelling of words,[32] but overfitting (i.e., training and memorising *too much* of the training data) is considered a bug since it reduces the ability of the model to make corrected predictions on new and unseen data.
- Besides overfitting, it has recently been shown that language models are prone to other forms of memorisation. This is a characteristic of modern deep learning that relates to instances such as interpolation[33], i.e., where the model perfectly fits all training labels.[34]

We posit that memorisation may play a key role in determining whether a model infringes copyright in the training data, such as in the case of literary works or images. One obvious example would be the case where there is literal reproduction, in the model and/or in the generated AI output, of text in the form expressed in the original data source. In the following section, we look more specifically at memorisation and explore its different forms which may play a pivotal role in copyright infringement analysis.

3. When is memorisation an issue?

[28] The idea is the following: we have a dataset of pairs of natural language and code; we pass each prompt to the model and for each prompt the model generates k code snippets; where the code snippet is correct, we say that the model has succeeded (i.e., passed) in k samples.
[29] ibid 2.
[30] Source code here should be understood, at most, as the model's weights. These, however, are not human-readable, so references to source code should be made with caution as there is not a relationship of identity between the source code of a computer program and model weights.
[31] Nicholas Carlini and others, 'Extracting Training Data from Large Language Models' (2021) 2634 <https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting> accessed 8 November 2022.
[32] ibid 2635.
[33] Interpolation refers to creating new data points within the range of known data points, much like 'best guessing'.
[34] Chen Cheng, John Duchi and Rohith Kuditipudi, 'Memorize to Generalize: On the Necessity of Interpolation in High Dimensional Linear Regression' (arXiv, 16 June 2022) 1–2 <http://arxiv.org/abs/2202.09889> accessed 10 May 2023.

It has been empirically shown that language models may exhibit behaviour that is similar to what we commonly known as photographic or eidetic memory, i.e., 'the ability to recall information after seeing it only once'[35]. In essence, eidetic memorisation refers to the behaviour of a model that memorises data when they appear only in a small set of training instances[36].

Eidetic memorisation is more likely to occur for data that does not appear as often in the training data, but may happen in other instances too. Memorisation is the opposite of generalisation.[37] In other words, if a model has memorised parts of its training data, then this means it has failed to generalise well. However, this type of memorisation is considered an error. Machine learning is premised on the idea of generalisation, since only generalisation allows the model to perform well when facing unknown data (e.g., the ability of ChatGPT to offer plausible answers).[38] Indeed, parameters are trained on data which sample a distribution and the model's behaviour should reflect that distribution. As discussed above, the purpose of the parameters and the embeddings derived from these parameters is not to encode the training data set but to represent information that can be induced from samples in the data set.

3.1. Sometimes memorisation is a feature, not a bug

A common mistake in non-technical literature is the assumption that all machine learning algorithms behave in the same way. This is of course a simplification, and one that has an impact on our analysis of generalisation vs memorisation. To illustrate, there are algorithms that create models which explicitly encode their training data, i.e., memorisation is an intended feature of the algorithm. These are, for instance, the k-nearest neighbour classification algorithm (KNN), which is basically a description of the dataset, and support vector machines (SVM), which include points from the dataset as 'support vectors'. Such models obviously memorise at least part of their training data.[39]

We are not particularly interested in KNN or SVM for several reasons, including the fact that they do not work well with large datasets or high dimensionality, and are particularly sensitive to noisy or mislabelled data. But it is interesting to signal that there are forms of machine learning where memorisation is the *intended* feature of the learning process. That said, our focus is on generative models built on the transformer architecture. In these cases, memorisation is undesired and is considered a bug. Nevertheless, as will be discussed in the following sections, it is plausible that copyright infringement is a *function* of memorisation.

3.2. Distinguishing unintended memorisation from overfitting

Before analysing the issues of eidetic memorisation in a legal context, we must first distinguish this phenomenon from **overfitting**. Overfitting is a well understood problem in deep learning. For example, it is much easier to create a machine learning classifier that perfectly fits its labelled training examples than one that can generalise on unseen data.

One way to understand overfitting is to compare the process with real-life learning in humans. If students study their class notes by heart without understanding the underlying concepts (i.e., without generalising), they may perform excellently on a test based exclusively on class notes material. However, when faced with an unknown problem, one where they need to prove their ability to apply their acquired knowledge to a different case, their performance drops significantly. Overfitting works in a similar fashion, and it is most often the result of overtraining the model, that is, training for too many times (epochs) on a single sample set of data[40]. The model will perform very well on the training data set or on data sets very similar to the training one, but will perform very poorly (i.e., will offer "wrong" answers) when confronted with unexpected environments, such as when a human asks a generative AI system to create a painting or compose a piece of software code. Reasons that may lead to undesirable overfitting can be found in small

[35] Carlini and others, 'Extracting Training Data from Large Language Models' (n 30) 2635, footnote 3.
[36] ibid.
[37] Xiaosen Zheng and Jing Jiang, 'An Empirical Study of Memorization in NLP' [2022] Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 2022 May 22 - 27 6265, 1.
[38] Cheng, Duchi and Kuditipudi (n 33) 1.
[39] Gavin Brown and others, 'When Is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning?', *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (2021) 1 <http://arxiv.org/abs/2012.06421> accessed 8 November 2022.
[40] Nicholas Carlini and others, 'The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks' (arXiv, 16 July 2019) 3 <http://arxiv.org/abs/1802.08232> accessed 28 July 2022.

training data size (i.e., not enough data samples), noisy data (i.e., containing a lot of irrelevant information), or high complexity where the model learns the noise within the data.

We measure overfitting by splitting the data into three categories: training, validation and testing data. The training data are used to train the classifier, the validation data are used to measure accuracy during the training, and testing data are used to assess accuracy after the training. Overfitting can be detected by measuring the training loss and the testing loss across the training or test inputs. Loss is a metric of how bad the model's prediction is on a single example. The lower the loss, the better the prediction. Empirical studies have demonstrated that the training loss decreases monotonically, but the validation loss decreases only initially[41]. When the model begins to overfit the training data, the validation loss increases, and the model becomes less generalisable[42]. The point where validation loss stops decreasing is the point of overtraining, as shown in Figure 5.

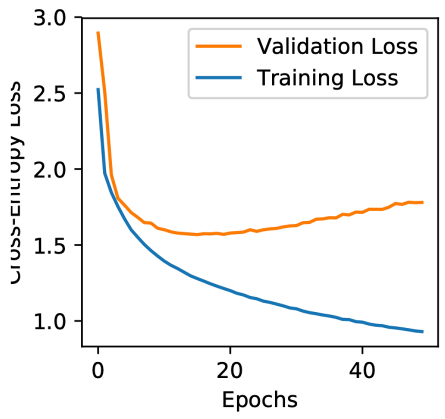

FIGURE 5. OVERFITTING AS A RESULT OF OVERTRAINING[43]

However, none of the memorisation examples given in literature are the result of overtraining.[44] If a model does not overfit its training data but still memorises parts of them, then there must be another explanation. We call these cases ***unintended*** eidetic memorisation. Unintended memorisation is not the result of overfitting, and it is not a feature of the selected algorithm. This type of memorisation cannot be eliminated by employing a different algorithm or by optimising the training process. One view suggests that unintended memorisation 'occurs when trained neural networks may reveal the presence of *out-of-distribution* training data—i.e., training data that is irrelevant to the learning task and definitely unhelpful to improving model accuracy'[45].

Three properties impact memorisation: **model scale**, **data duplication** and **context**. In terms of scale, larger models are known to memorise more data[46]. Furthermore, the more often sequences are repeated in the training data set, the more likely the model is to memorise them. Even deduplication is of limited utility as memorisation could occur with even a few duplicates[47]. Additionally, models may only exhibit their propensity to memorise when prompted with a longer context. This makes discovering memorisation even more challenging[48]. Unintended memorisation is thus said to 'memorize rare details about the training data that are completely unrelated to the intended task while the model is still learning the underlying behaviour'[49].

[41] ibid.
[42] ibid.
[43] ibid.
[44] ibid.
[45] ibid.
[46] Nicholas Carlini and others, 'Quantifying Memorization Across Neural Language Models' (arXiv, 15 February 2022) 5 <http://arxiv.org/abs/2202.07646> accessed 28 July 2022.
[47] ibid 5–6.
[48] ibid 6.
[49] Carlini and others, 'The Secret Sharer' (n 39) 16.

Memorisation, however, is not always the result of a model that performs poorly. In fact, it has been theorised that memorisation may be beneficial to generalisation. One such view draws on the long-tailed nature of data distributions. It states that memorisation of useless examples and the ensuing generalisation gap is necessary to achieve close-to-optimal generalisation error[50]. Allegedly, this happens when the data distribution is long-tailed, that is, when rare and non-typical instances make up a large portion of it. In long-tailed distributions, useful examples, which improve the generalisation error, can be statistically indistinguishable from useless examples, which can be outliers or mislabelled examples.

The long-tail theory has been tested in experiments on natural language processing tasks such as sentiment classification, natural language inference and text classification. Specifically, it has been shown that the 'training instances with the highest memorization scores are atypical' and that 'removing the top-memorized training instances results in significantly dropped performance [which is] markedly higher compared with removing a random subset of training instances'[51]. This suggests there may be a trade-off between a model's accuracy and the memorisation of data[52]. In other words, it seems that memorisation can be beneficial to and, in fact, needed for generalisation when the data distribution has a long-tail nature. Long-tailed distributions are typical in many critical machine learning applications for face recognition, age classification and medical imaging tasks.

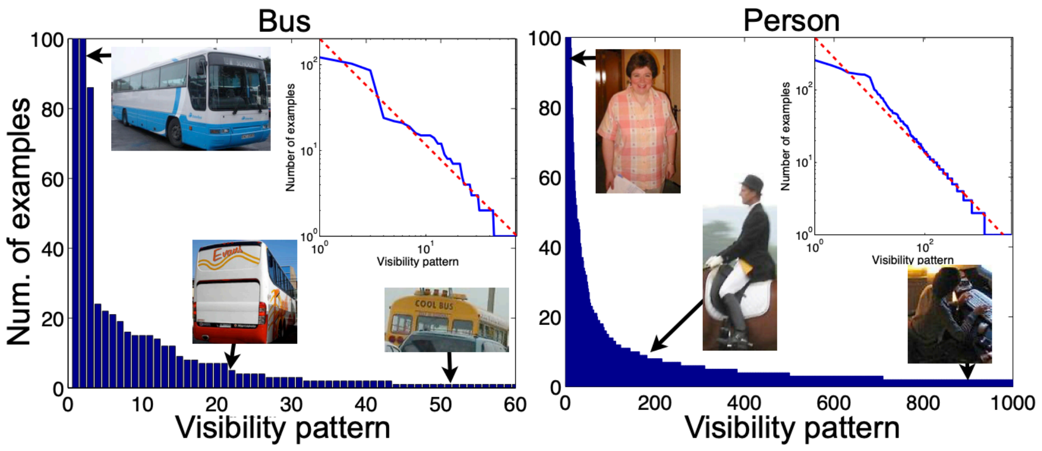

FIGURE 6. LONG-TAIL DISTRIBUTIONS OF THE VISIBILITY PATTERNS FOR BUS AND PERSON[53]

We summarise our findings of the different types of memorisations in the following table:

TABLE 2. CLASSIFICATION OF MEMORISATION

MEMORISATION TYPE	CAUSE	EFFECT
INTENDED MEMORISATION (ALGORITHM DESIGN)		
Memory-based learning (eg, k-Nearest Neighbours (kNN) algorithm)	The algorithm replaces model creation by memorising the training data set and then use this data to make predictions.	Encodes part of the data set. Highly accurate predictions but poor performance on large datasets or high dimensionality.

[50] Vitaly Feldman, 'Does Learning Require Memorization? A Short Tale about a Long Tail', *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery 2020) 955 <http://doi.org/10.1145/3357713.3384290> accessed 8 November 2022; Vitaly Feldman and Chiyuan Zhang, 'What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation' (arXiv, 9 August 2020) 2 <http://arxiv.org/abs/2008.03703> accessed 18 November 2022.
[51] Zheng and Jiang (n 36) 2.
[52] ibid 9.
[53] Xiangxin Zhu, Dragomir Anguelov and Deva Ramanan, 'Capturing Long-Tail Distributions of Object Subcategories', *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014) 1 <https://ieeexplore.ieee.org/document/6909517> accessed 24 October 2023.

Support Vector Machines (SVM)	The algorithm uses a subset of training points from the dataset in the decision function (support vectors)	Encodes part of the dataset as support vectors. Effective in high dimensional spaces.
UNINTENDED MEMORISATION THAT DOES NOT IMPROVE ACCURACY		
Overfitting	Most often the result of overtraining the model, ie, training for too many times on a single sample set of data.	Contains more parameters than can be justified by the data (overparametrised). Model performs very well on the training data set or on data sets very similar to the training one but will perform very poorly when confronted with unexpected environments.
Eidetic memorisation	Trained neural networks encode out-of-distribution training data, ie, training data that is irrelevant to the learning task and unhelpful to improving model accuracy.	Encodes parts of the dataset. Model does not perform better.
UNINTENDED MEMORISATION NECESSARY FOR GENERALIZATION		
Long-tailed memorisation	Memorisation of useless examples and the ensuing generalization gap is necessary to achieve close-to-optimal generalization error. Occurs in long-tail data distributions a significant fraction of which are made up of rare and atypical instances.	Encodes parts of the datasets (label memorisation). Memorisation can be beneficial to and, in fact, needed for generalisation when the data distribution has a long-tail nature.

If memorisation, particularly unintended memorisation, may be necessary for models to generalise better in certain tasks, particularly in cases of large models like LLMs, then this phenomenon may have broader legal implications when these models are distributed or used to generate code.

The following section explores the implications of memorisation for copyright analysis in the context of large language models for code and services implementing these models, such as GitHub Copilot.

4. MEMORISATION OF SOURCE CODE THROUGH THE LENS OF COPYRIGHT INFRINGEMENT

Memorisation in large language models does not seem to have garnered the attention of copyright lawyers. From a legal perspective, memorisation has been studied primarily in the domain of machine learning privacy.[54] Naturally, models that can reproduce personal data about individuals raise concerns from a data protection and privacy perspective.[55] In this context, it has been argued that although memorisation can reduce the generalisation error, i.e., reduces inaccuracy in predicting outcome values on unseen data, it could also increase the model's vulnerability to

[54] Carlini and others, 'The Secret Sharer' (n 39); Carlini and others, 'Extracting Training Data from Large Language Models' (n 30); Carlini and others, 'The Secret Sharer' (n 39); Nicholas Carlini and others, 'The Privacy Onion Effect: Memorization Is Relative' (arXiv, 22 June 2022) <http://arxiv.org/abs/2206.10469> accessed 8 November 2022; Brown and others (n 9); Brown and others (n 38); Zheng and Jiang (n 36).

[55] Michael Veale, Reuben Binns and Lilian Edwards, 'Algorithms That Remember: Model Inversion Attacks and Data Protection Law' (2018) 376 Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 20180083.

privacy attacks[56]. We argue that a similar trade-off can be seen between the accuracy of a model and its potential to infringe on copyright.

The copyright implications of memorisation have only recently become subject of debate in the research community. For example, prompting GPT-3 with the beginning of a chapter of JK Rowling's *Harry Potter and the Philosopher's Stone* has been shown to trigger the model to reproduce verbatim a full page of the chapter before it makes a mistake[57]. Similarly, GitHub Copilot was shown to reproduce verbatim copies of well-known code from the computer game Quake III[58] as well as code belonging to individual programmers.[59]

These allegations have prompted a group of class-action lawyers to file a lawsuit in the US District Court for the Northern District of California on behalf of a proposed class of potentially millions of GitHub users.[60] In the class-action complaint, they challenge the legality of GitHub Copilot and its underlying model Codex on grounds of violating the rights of developers who have uploaded code and other works to GitHub under a set of 11 popular free and open source software licenses, such as the MIT, Apache and GPLv2/v3.

4.1. A DEVELOPER'S VIEW

Codex, the language model underlying the GitHub Copilot service, was trained by OpenAI and GitHub, both of which are companies headquartered in the US. It is therefore reasonable to assume that its authors have relied on US copyright law, particularly fair use, to perform this activity. Indeed, in their paper documenting the model, they clearly state that 'the training of AI systems on Internet data, such as public GitHub repositories, has previously been identified as an instance of "fair use"'[61]. This assumption seems to largely rely on one of the most prominent cases in the US that comes close to the factual circumstances of training of machine learning models, namely the *Authors Guild v Google, Inc* case[62]. In this case, activities such as mining the web and creating thumbnails from book pages were considered fair use due to their transformative nature. It is important to note, however, that the US Supreme Court refused to hear the case. The holding of the 2nd circuit therefore stands as an influential precedent but may not be the final word in this account.[63]

While this paper does not discuss the US fair use doctrine, a few cautionary words may be relevant to this analysis. The argument that model training is an instance of fair use may be at odds with the idea that models may memorise their training data and reproduce them verbatim. The facts in the *Authors Guild* case concerned searches and thumbnailing, i.e., forms of transformative use of the underlying works. This aspect could certainly be equated to the case of machine learning and the underlying training data.[64] However, memorisation, i.e., the making of verbatim copies of parts of the (copyright protected) training data, may not follow the logic of the court in their application of the transformative condition. Google made copies of books not to display them but to allow users to search and see snippets in context. This added value of the service is beyond what a user could get from individually owned copies. Furthermore, Google did not aim to offer substitutes in the market for books, but to offer a different, albeit connected, service. In contrast, Codex is able to reproduce full implementations of original program code, thereby creating a

[56] Zheng and Jiang (n 36) 9.

[57] Eric Wallace and others, 'Does GPT-2 Know Your Phone Number?' (*The Berkeley Artificial Intelligence Research Blog*, 20 December 2020) <http://bair.berkeley.edu/blog/2020/12/20/lmmem/> accessed 19 December 2022.

[58] Stefan Karpinski [@StefanKarpinski], 'In Case It's Not Clear What's Happening Here: @github's Copilot "Autocompletes" the Fast Inverse Square Root Implementation from Quake III — Which Is GPL2+ Code. It Then Autocompletes a BSD2 License Comment (with the Wrong Copyright Holder). This Is Fine.' <https://twitter.com/StefanKarpinski/status/1410971061181681674> accessed 26 October 2023.

[59] Ryan J Salva, 'Preview: Referencing Public Code in GitHub Copilot' (*The GitHub Blog*, 1 November 2022) <https://github.blog/2022-11-01-preview-referencing-public-code-in-github-copilot/> accessed 15 May 2023.

[60] Butterick and others (n 5).

[61] Chen and others (n 23) 13.

[62] *Authors Guild v Google, Inc* (2014) 804 F 3d 202 (Court of Appeals, 2nd Circuit).

[63] Bradley Kuhn, 'If Software Is My Copilot, Who Programmed My Software?', *Free Software Foundation* (2022) 4; *Maryland v Baltimore Radio Show, Inc,* (1950) 338 338 US 912 912 (US Supreme Court).

[64] Kuhn (n 62) 4.

substitution effect in the market which the underlying works are intended to serve. Even the authors of Codex explicitly acknowledge the possible economic and labour market impacts of their model.[65]

On the other hand, code is brittle, big and diverse, and needs more precise outputs compared to natural language.[66] Codex cannot really handle well typical daily problems in software engineering, such as the ability to scale up. Additionally, neither Codex, nor any other large language model can deal with locality, i.e., it does not know where to place the code it has just synthesised, for example, in a modular application. Finally, Codex cannot address the issue of intent, i.e., to be able to infer with precision from the program specifications the functionality intended by the programmer.[67] Think of the more familiar (and far less complex) auto-complete function of a word processor: it may be very handy for regular, short and common expressions, but it often fails to properly complete more complex, long and original ones, especially when the context of the paragraph or chapter is essential to determine the specific expression. These issues raise the question of whether Codex or any other generative model can benefit from a finding of fair use in the same way in which, illustratively, mining the web has benefitted Google.

The fair use exception may thus be a much more uncertain ground than initially portrayed by machine learning developers.[68] This is especially so in cases where the model output can be seen as a substitute to the training data, as is arguably the case with so-called generative AI. Using fair use as a legal basis to justify the unrestricted training of a large language model on 'public Internet data' is not a fully settled question, and distinctions need to be made between the input and output phase.

In infringement proceedings, memorisation can play a decisive role. A balancing exercise will need to consider the impact of model outputs in relation to the market served by the training sources. The impact on the development of a critical technology like AI, both domestically and globally, will likewise play a role in determining where to draw the line between lawful and infringing uses, or, in normative terms, in defining the precise scope of copyright in the age of machine learning.

4.2. Overview of the EU copyright framework

Getting back to the central problem of this paper, we need to understand how the EU legal framework applies to acts of mining and training of protected works, as well as how the use of open-source licenses changes the outcome of this analysis. In trying to answer these questions, we use memorisation as a benchmark for interpretative and normative guidance.

On the question how mining and/or training activities are classed by EU copyright law, and whether they require authorisation, the answer has been discussed, as least partly, both at statutory and judicial level, as well as in scholarship.[69] For the purposes of our analysis, it is sufficient to synthesise the main elements. The case law of the Court of Justice of the EU (CJEU) adopts a rather formalistic approach to maintain that where technological processes require the making of copies within the broad definition of Art 2 InfoSoc Directive, which includes also temporary and incidental copies, an authorisation (statutory or contractual) is normally required. The qualifier "normally" is warranted due to a recent and interesting development in CJEU copyright case law in relation to the delimitation of the right of reproduction for neighbouring rights. In *Pelham* case,[70] the CJEU established that, in principle, even short

[65] This also alludes to the advertisement of the related GitHub Copilot service which presents the plugin as a 'pair programmer'.
[66] Armando Solar-Lezama, 'LLMs for Code' (Generative AI: Language, Images and Code - a conversation with CSAIL 2023, Cambridge, MA, USA, 10 April 2023) <https://www.youtube.com/watch?v=VDKkc36lGeg> accessed 24 October 2023.
[67] ibid.
[68] Benjamin LW Sobel, 'Artificial Intelligence's Fair Use Crisis' (2018) 41 The Columbia Journal of Law & the Arts 45, 75.
[69] Thomas Margoni and Martin Kretschmer, 'A Deeper Look into the EU Text and Data Mining Exceptions: Harmonisation, Data Ownership, and the Future of Technology' (Zenodo 2021) <https://zenodo.org/record/5082012> accessed 6 July 2022; SEAN M Fiil-Flynn and others, 'Legal Reform to Enhance Global Text and Data Mining Research' (2022) 378 Science 951.
[70] *Pelham GmbH, Moses Pelham, Martin Haas v Ralf Hütter, Florian Schneider-Esleben* [2019] Court of Justice of the European Union C-476/17, ECLI:EU:C:2019:624.

reproductions of sound recordings (e.g., 2 seconds) may constitute an infringement of the related right in phonograms (Art 2(c) InfoSoc). The Court held, however, that:

"[W]here a user, in exercising the freedom of the arts, takes a sound sample from a phonogram in order to use it, in a modified form unrecognisable to the ear, in a new work, it must be held that such use does not constitute 'reproduction' within the meaning of Article 2(c) of Directive 2001/29"[71]

The Court seemingly associates the 'recognisability' element with a balancing exercise between fundamental rights (i.e., freedom of artistic expression) and the justifications underpinning the related right in phonograms, namely the protection of the investment. This is an interesting development, although it leaves several questions open. It is not clear how to assess whether a sample is recognisable and, importantly, to whose ear. Perhaps even more crucial, though, is the question whether this turn in the infringement test, which arguably introduces a limitation for transformative uses in EU law, extends also to Art 2(a) InfoSoc Directive, i.e., to the right of reproduction for *works* and not only for phonograms.[72] Given the extremely wide scope of the right of reproduction under EU law, this would be a welcome development, but one that needs to be taken with a grain of salt. On one hand, the previously consistent line of cases on the right of reproduction for works (Infopaq et al[73]) consistently supports the formalistic approach described above. On the other hand, a series of factors weight against extending the ratio of the ruling in *Pelham* to the right of reproduction for works. It is undeniable that there are many specific references in the *Pelham* decision to the fact that the right at issue was a *related* right to copyright and that this right is justified not by the protection of the intellectual creation but by the protection of the investment of the producers which is not compromised by the new test. It seems safe to assume that, under current law, recognisability has not yet found its way in the copyright (as opposed to neighbouring rights) case law of the CJEU.

Whether "recognisability" is going to become the new infringement benchmark for other neighbouring rights – perhaps even including the *sui generis* database right – is also unclear. At any rate, it would certainly represent a logical expansion of the rationale underpinning the court's findings in its analysis of the production and commercialisation of phonograms. Accordingly, the expansion to other neighbouring rights would appear as the possible next step of the newly born recognisability test in EU related rights.

Beyond the above considerations on infringement – and thus ultimately on copyright's scope – an authorisation, statutory or contractual, for the use of works is required. Statutory forms of authorisation can be found in the current *acquis* on exceptions and limitations.[74] For technological processes, Article 5(1) InfoSoc Directive[75] on temporary copies has traditionally operated as a balancing interface between the protective effect for right-holders of a broad right of reproduction (Article 2 InfoSoc Directive) and the pro innovation and consumer welfare effects of favouring technological development.[76] This position is supported by the case law of the CJEU in a variety of cases, perhaps the most factually relevant of them being the search, retrieval and extraction of keywords in a news information service.[77]

Other provisions similarly offer a form of a statutory authorisation in specific cases, including for research and teaching (Article 5.3.a InfoSoc Directive[78]), which, it is worth reminding, was the basis for the creation of domestic

[71] ibid 31.
[72] Ironically the same Court's holding explicitly denied the compatibility with EU law of the more or less equivalent statutory German concept of free uses.
[73] *Infopaq International A/S v Danske Dagblades Forening* [2009] ECJ Case C-5/08, ECLI:EU:C:2009:465.
[74] For licenses see infra.
[75] Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society 2001 art 5(1).
[76] *Infopaq International A/S v Danske Dagblades Forening* (n 72); *Football Association Premier League and Others* [2011] Court of Justice of the European Union C-403/08, ECLI:EU:C:2011:631.
[77] *Football Association Premier League and Others* (n 75); *Infopaq International A/S v Danske Dagblades Forening* (n 72).
[78] Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society art 5(3)(a).

text and data mining exceptions prior to the adoption of the 2019 CDSM Directive. Nevertheless, Articles 3 and 4 of the CDSM Directive[79] reflect the current law in the EU in so far as text and data mining is concerned.[80]

4.3. Arts 3 and 4 CDSM Directive: selected aspects

As discussed in more detail elsewhere[81], Art 3 CDSM Directive is available to research organisations, cultural heritage institutions and certain other entities operating for research purposes. For-profit text and data mining, such as in the discussed case of GitHub Copilot, is therefore outside the scope of Art 3. Relevant in this case is Art 4 CDSM Directive which exempts text and data mining acts carried out by any beneficiary and for any purpose, including for profit. It is this provision that may cover commercial generative AI development.

There are two main elements in Art 4 that merit further attention. The first is that Art 4 CDSM Directive is subject to a reservation by rightsholders (section 4.3.1). The second is that Article 4 CDSM Directive explicitly covers the right of adaptation for computer programs (section 4.3.2), but not of other copyright subject matter. Finally, there is the question whether Arts 3 and 4 CDSM Directive cover the current use of models trained *prior* to the entry into force of these provisions.[82]

4.3.1. Opt-out under Article 4 CDSM Directive

The text of Art 4(3) CDSM Directive states that the TDM exception only applies on condition that the use of the work has not been expressly reserved by its rightsholders in an appropriate manner, such as machine-readable means in the case of content made publicly available online. Recital 18 clarifies further that in the case of content that has been made publicly available online, it should only be considered appropriate to reserve those rights using machine-readable means, including metadata and terms and conditions of a website or a service.

The requirement of an "express" reservation seems to point in the direction of a clear reference to text and data mining uses. To illustrate, a hypothetical *TDM.txt* file could meet these requirements, whereas a more general *robot.txt* may be too generic to trigger this effect.[83] By the same token, other common statements such as "all rights reserved" or "only for personal non-commercial uses"– assuming they are expressed in machine readable formats – do not appear to possess the required degree of precision to trigger the effect of Art. 4(3) CDSM Directive. Whether an *AI.txt* statement is sufficiently "express" is probably a function of whether EU TDM exceptions cover AI training. If they do, which is the predominant view, then and *AI.txt* reservation should be express enough.

When the training material used for text and data mining is released under free and open-source software licences or other open content licences like the Creative Commons family of licences, the training, including for commercial purposes, is generally allowed under the provision of Art 4, making the license conditions irrelevant. In fact, these licenses commonly indicate that they are "triggered" only if applicable law restricts a certain use. As long as applicable law permits a certain use (for instance because it is not restricted by copyright or it is allowed under an exception), the license does not matter. Accordingly, as long as AI training is done within the boundaries of Art 4, the clauses and conditions of most floss or open content licences do not matter. More explicitly, in the case of a CC 4.0 BY-NC which limits the license grant to non-commercial uses, to use such content for model training, including for commercial

[79] Directive (EU) 2019/790 of the European Parliament and of the Council of 17 April 2019 on copyright and related rights in the Digital Single Market and amending Directives 96/9/EC and 2001/29/EC 2019 (OJ L) arts 3, 4.

[80] It appears unlikely that in the rather formalistic framework of EU copyright law normative doctrines such as that of non-consumptive uses may find fertile ground despite offering a potentially well-balanced answer.

[81] Margoni and Kretschmer (n 68) 694. Kretschmer et al, 2024. Copyright law, and the lifecycle of machine learning models. International Review of Intellectual Property and Competition Law (IIC); 2024; Vol. 1/2024; iss. 1, forthcoming.

[82] Especially given that Arts 3 and 4 CDSM Directive need transposition into the national law of EU Member States – a process that started only in 2019 and has not been completed in some Member States yet. This issue will not be explored in further details here.

[83] We use a hypothetical TDM.txt example assuming that it will contain a machine-readable string that refers specifically to the opting out from TDM activities. Therefore, we do not refer to any existing initiative specifically. However, the AI.txt initiative developed by Spawning.AI, provided it possesses the requirements of an explicit reservation, could certainly represent a promising technological measure under the framework laid out by Art 4 CDSM Directive.

purposes, is allowed, as long as covered (in the EU) by Art 4 CDSMD. This is because, if an exception applies the licence is not triggered.[84] The proposition that clauses like "Non-Commercial" or "Non-Derivatives" could be seen as a form of Art 4 reservation is simply untenable. Not only would this be too generic to qualify as an Art 4 "express" reservation. This view would also conflict with the spirit and letter of open licences which are intended to offer more generous re-use rights than copyright's default regime.[85] Thus, it seems safe to assume that open-source software licences and Creative Commons public licences (CCPL) cannot be construed as an Art 4 opt-out.

In the slightly illogical but technically possible situation of the application of an express reservation under Art 4 CDSM Directive to a CC-BY 4.0 or GPLv3 licensed material, two interpretations are possible. The first suggests an incompatibility between Art 4 opt-out and the use of an open-source software or open content licence. This view is predicated on the fact that the spirit and letter of open-source software and open content licences are to offer more generous access and reuse rights than copyright's default rules. Opting out in these cases would appear as inconsistent and contradictory behaviour on the part of the licensor, up to the point of *venire contra factum proprium*, and therefore usually rejected by legal systems based on a variety of legal doctrines (from *bona fide* remedies to different forms of estoppel). This interpretation clearly favours licensees and innovation by lowering the barriers to reuse.

The second possibility is that Art 4 CDSM Directive offers right-holders (and prospective licensors) the possibility to crystallise the applicable copyright rule in this specific area. Once right holders/licensors manifest their power to reserve or not text and data mining under Art 4, then copyright's final shape in relation to this specific transaction is set. At this point, an open-source software or open content licenses will apply to the licensed work in light of the applicable copyright law. In this case, it would be possible for a right holder to reserve the right of text and data mining (other than those included in Art 3) via an Art 4 opt-out and then allow text and data mining on the basis of the conditions of the chosen license. These may include, for instance, obligations to maintain a copyright notice, attribute authorship, apply the same license conditions (e.g., GPLv3 or CCPL BY-SA 4.0), reserve commercial uses (CCPL BY-NC 4.0) or forbid the creation of derivative works (CCPL BY-ND 4.0). This second interpretation admittedly favours licensors and allows them a more granular control of the possible uses of their works for text and data mining and/or generative AI applications. Naturally, the licence conditions only apply to the uses that are stipulated in the licences. Depending on which licences and which specific type of use is at stake, it is possible that licence conditions may not be triggered in cases of mere mining, but they may be triggered in situations where the output is seen as a form of distribution or derivation of the original training material.

4.3.2. Software and databases

Regarding the explicit inclusion of computer programs in Art 4 CDSM Directive, it is necessary to briefly analyse the implications of its presence in Art 4 and of its absence in Art 3 CDSM Directive. As we shall see, this assessment also extends to databases.

The acts exempted by Arts 3 and 4 CDSM Directive are very similar but not identical. Art 4, in addition to the acts covered by Art 3 CDSM Directive, specifically refers to Art 4(1)(a) and (b) of Directive 2009/24/EC.[86] The reference is to the Computer Programs Directive and, specifically, to the right of reproduction and the right of adaptation. It is not fully clear why Art 3 does not refer specifically to the Computer Programs Directive, while it refers, for instance, to the Database Directive (DD). It is also not clear why the adaptation of computer programs is included in the scope of Art 4 CDSM Directive, while the equivalent adaptation right of databases (Art 5(b) DD referring to copyright) is not.[87] It should be reminded that, barring specific cases such as computer programs and databases, the adaptation right in EU copyright law has not received horizontal harmonisation.

Regarding the DD, it should be further noted that, despite employing a rather ambiguous formulation, it could at least be argued that both Arts 3 and 4 CDSM Directive, in relation to the *sui generis* database right (SGDR), cover not only acts of reproduction ("extraction") but also acts of communication to the public, distribution, renting and any other

[84] E.g., see Sec. 2.a.2 CC 4.0 BY-NC available at: https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode.en.
[85] For CCPL BY 4.0, see, for instance, the Preamble, Sections 6.c, 7 and 8.a; for GPLv3, see the Preamble.
[86] Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs (Codified version) 2009.
[87] Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases 1996 (OJ L) art 5(b).

form of transmission ("reutilisation"). The reason is that Arts 3 and 4 are exceptions to Art 7(1) DD which refers both to extractions and reutilisations.[88]

The ambiguity stems from the specific wording of Arts 3 and 4 CDSM Directive. Paraphrasing them and substituting the legislative references with the provisions contained in Art 7(1) DD, they would read: "Member States shall provide an exception to the right of *reproduction*, *communication to the public* and *distribution* for *reproductions* made for the purpose of scientific research". It could be argued that the reference contained in Arts 3 and 4 CDSM Directive to Art 7(1) DD is simply imprecise. It should have been a reference only to the first part of Art 7(1), thus only to extractions and not to reutilisations and that this is reflected in the part of Arts 3 and 4 that refer only to *reproductions*. This is plausible, although it deviates quite substantially from the literal meaning of the provision. It is also in sharp contrast with the precision of the legislative references in the rest of the provisions.

An alternative plausible reading would suggest that if an extraction has been made for the identified purpose, the extracted content protected by SGDR can be further distributed, made available to the public and otherwise transmitted. Logically, this would only apply to the content of the database (protected by SGDR) and not to its structure (protected by copyright), but this could be in line with the technical nature of mining database content. The provision remains nevertheless vague.

Moving to the issue of software, a likely explanation of its presence in Art 4 CDSM Directive (or absence in Art 3) could be found in the last-minute addition of Art 4 in the legislative process reducing the opportunities for commentators to intervene or to investigate the legislative history. Regardless of the genesis of the rule, it is beyond any doubt that Art 4 CDSM Directive covers not only reproductions but also adaptations of computer programs. This has interesting repercussions on cases like GitHub Copilot.

On the other hand, the absence of any specific reference to the Computer Programs Directive in Art 3 CDSM Directive raises the question whether it is possible at all to mine software under Article 3 CDSM Directive. One could argue the right of reproduction covered in Art 3 CDSM Directive is broad enough (as *lex generalis*) to include the not explicitly mentioned right of reproduction for software under Article 4 CDSM Directive (*lex specialis*). The possibility to mine software on the basis of Article 5(3) Computer Programs Directive is a safer, yet imperfect, solution.[89] One explanation for the difference between Arts 3 and 4 CDSM Directive regarding software is that the legislator has focused mainly on scientific publishers and research organisations at a time when software mining was not on the legislative agenda.

4.3.3. Do Arts 3 and 4 CDSM Directive cover generative AI?

An additional important but often ignored aspect is whether the text and data mining rules apply also to so-called generative models. When performing text and data mining, the results that are normally obtained – at least from the point of view of the law – include "patterns, trends and correlations".[90] Generative AI often shares with text and data mining many of the technological steps that constitute relevant legal acts. However, generative AI can produce far more advanced outputs, such as literary or artistic content of a quality that plausibly looks like works created by humans, i.e., synthetic content. It is precisely this feature of generative models that has contributed to its negative image in many the creative sectors.

Art 2 CDSM Directive offers a broad characterisation of text and data mining and defines it as "… any automated analytical technique aimed at analysing text and data in digital form in order **to generate information** which ***includes but is not limited to*** patterns, trends and correlations".[91] At the time of drafting (before 2019) this broad definition could probably not have been fully informed by the recent advances in AI. However, a literal interpretation seems to offer enough ground to cover the process of *generation* of information. The examples of what counts as "information"

[88] ibid 7(1) and (2).
[89] Rossana Ducato and Alain M Strowel, 'Ensuring Text and Data Mining: Remaining Issues With the EU Copyright Exceptions and Possible Ways Out' (1 February 2021) <https://papers.ssrn.com/abstract=3829858> accessed 26 October 2023.
[90] Copyright in the Digital Single Market Directive art 2.
[91] ibid 2(2).

under Articles 3 and 4 CDSM include patterns, trend and correlations, but the enumeration is not a *numerus clausus* (“*includes but is not limited to*”), so the list of eligible ‘information’ is open-ended and thus capable of covering other connected uses, such as generative AI. Additionally, the AI Act proposal, European Parliament text, appears to confirm the view that Art 4 CDSMD can be used as a legal basis for generative AI by requiring a disclosure of the training material covered by copyright, a provision that seems directly connected to the reservation of rights and the ensuing licensing opportunities.[92]

Including generative AI in the scope of Arts 3 and 4 CDSM Directive has the incontestable advantage of offering a solution that has already been the subject of a recent balancing exercise made of public consultations, impact assessments and adopted legislation. One may like or dislike the degree of this balance as a matter of copyright policy, but it is a balance nonetheless, and one that offers certain clearly excluded areas (Art 3), other areas where right holders can reserve the right of text and data mining (Art 4), and finally yet other areas that are automatically reserved to right holders (activities outside the scope of Arts 3 and 4, e.g., communication to the public). Additionally, Art 2’s broad definition of text and data mining is already capable of harbouring many activities that go under the name of generative AI. It is therefore more likely than not that Arts 3 and 4 will likely be, and in fact already are, the regulatory mechanisms for AI development under EU copyright law.

4.4. OPEN-SOURCE SOFTWARE AND OPEN CONTENT LICENCES

Whether acts of training are covered by the provisions of open source software or other open content licences is an interesting question, but it is one that has yet to be addressed definitively by courts. As a general principle, open source licences normally only require compliance with their conditions if a use would be otherwise reserved under applicable (copyright) law.[93] Furthermore, in most cases, the licence conditions are triggered only upon distribution. If an act of training is covered by an exception or limitation to copyright or is not a copyright relevant activity in the first place, it is not necessary to explore further the conditions of the underlying licence. The legal basis for use would clearly be a statutory exemption, not a licence.

If exceptions and limitations do not apply, it becomes necessary to analyse the conditions of individual licenses. While a licence specific analysis exceeds the scope of our article, it suffices here to say that under most open source software licenses, as well as Creative Commons licenses (for instance the CC BY 4.0),[94] permit the acts that from a legal perspective constitute “training”, of course under the conditions established in the licence. This is so because training often consists, from a copyright point of view, mainly of temporary and/or permanent reproductions, and depending on the specific environment/application, sometimes of communications to the public and adaptations. Licences such as CC BY 4.0 expressly allow all these activities for rights included in copyright and related rights’ scope, such as the SGDR. Naturally, if the legal basis for training is the licence, the licence conditions must be observed. Thus, in the case of a CC BY 4.0, mainly the preservation of the licence and the recognition of authorship in a reasonable manner.[95] This situation appears well accepted in practice in the field of LLMs where CC-BY content, such as articles on Wikipedia, features prominently in the training datasets, whenever these are disclosed.

In the absence of any more precise judicial guidance, it seems plausible that open source software licences operate in a similar fashion. Practice shows that code released under these licences is a popular choice for machine learning, as exemplified by the use of GitHub repositories by Copilot. Where copyright restricts training acts, then training can take place on the basis of the specific licence.

It is interesting to note that the mere ‘use’ of the source code for training purposes would not have to comply with the licence conditions because for most licences these conditions are triggered only upon distribution. For example, the licence conditions of the GPLv3 are triggered by the act of conveyance (Sections 4-6). The GPLv3 defines the act of

[92] See Amendment 399 “Proposal for a regulation Article 28 b (new)” of the Amendments adopted by the European Parliament on 14 June 2023 on the proposal for a regulation of the European Parliament and of the Council on laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts (COM(2021)0206 – C9-0146/2021 – 2021/0106(COD).

[93] See, eg, Section 2(a)(2) CCPL BY 4.0; Section 2 GPLv3.

[94] This are licenses that do not conditioning derivatives to the same license)

[95] ‘Content Mining - Creative Commons’ <https://wiki.creativecommons.org/wiki/Content_mining> accessed 15 May 2023.

conveying a work to mean "any kind of propagation that enables other parties to make or receive copies"[96]. Propagation includes copying, distribution (with or without modification), and making available to the public. As a rule of thumb, if there is no redistribution of the program (i.e., any copyrightable work licensed under the GPLv3), the licence does not require to meet the conditions. Indeed, the GPLv3 Section 2 is explicit in providing that one may make, run and propagate covered works that are not conveyed, without conditions so long as your license otherwise remains in force. The GPLv3 makes it clear that cases of use of a program in Software-as-a-Service scenarios do not trigger the licence conditions. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.[97] In other words, where an application is accessed through a web browser and no client-side software involves a transfer of a GPLv3-licensed code, then the provider does not need to meet the requirements of the licence vis-à-vis recipient(s) of the service. This of course makes sense, and it is the crucial distinction between a licence agreement and a service agreement that is often neglected in practice. Where distribution or conveyance occurs, the distributor would be obliged to maintain the copyright notice – a feature common to most open-source software licences which includes a credit/authorship notice – and, for reciprocal licences, to release under the same conditions.

In the cases where the trained model is seen as a reproduction (covered in both Art 3 and 4 CDSM Directive) or an adaptation (covered only for computer programs in Art 4 CDSM Directive) of the training data, that model cannot be distributed or communicated to the public on the basis of these two articles. They could, however, be shared based on a specific licence permission (e.g., CCPL or GPLv3) under the same conditions highlighted above. More importantly, it seems that, at least in the ways that we can observe right now, trained models are not distributed or communicated to the public but run on the cloud servers of companies which give access to their services usually through private application programming interfaces (APIs). This type of business model, which at least for now seems to characterise most implementations of commercial generative AI tools, will probably sidestep the issue of communication to the public. However, the same does not apply to the output of a generative model, which, by definition, must be communicated to the public (the answer that ChatGPT returns when you ask something needs to reach you). If the generated output is a reproduction in part or an adaptation of the training material, Arts 3 and 4 CDSM Directive are of little help, and the licence conditions of the original training material will determine the answer to this question. The compatibility of these licenses, particularly in cases of adaptation/derivative works, is yet another aspect that must be addressed.

4.5. Remaining issues: input data and creative adaptations

Before moving to the next section, we should point out that our analysis has focused on, and is limited to, the role of training data. The role of input data, i.e., the data inserted by a user interacting with the trained model (i.e., the question asked by the user), raises additional issues. It cannot be excluded that, at least in certain specific cases, the "question" asked by the user is original enough to be copyright eligible in its own right. Accordingly, it cannot be excluded, at least theoretically, that in similar cases, the results generated by the model may be a reproduction and/or adaptation of both the training data and the user input data, raising interesting issues of complex adaptations and/or joint works. It seems that, regarding user input data, in most cases their use is governed by the terms of use of the user-facing application.

Whether from an EU law point of view trained models or the generated output can be considered adaptations or derivative works is a very interesting question, which once again cannot be fully explored in this paper. Two aspects can, however, be pointed out. The first is that the answer, even within the EU, may vary from one Member State to the other since the right of adaptation is not horizontally harmonised. Secondly, the Computer Programs and the Database Directives have harmonised the adaptation right for their respective subject matter only. Thus, the answer may not only vary depending on the Member States but also depending on whether the training material, the trained model and the generated output are generally classified as works of authorship or whether they belong to the two aforementioned special subject matter categories.

[96] GPVv3, Section 0 Definitions.
[97] Pursuant to GPLv3 Section 0 'Definitions', mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

4.6. Has Codex memorised its training data?

The training data set of Codex consists of computer programs that are protected as literary works under copyright law. By now it should be clear that (parts of) copyrighted works may be memorised during the training process. When such 'big code' models are used to generate 'new' code, developers may be faced with a situation where their generated code could turn out to be an infringing work if it includes verbatim portions of the original training code.

The developers of Codex claim that 'preliminary research … finds that Codex models rarely generate code that is identical to the contents of training data' with 'occurrences … <0.1% in a study examining the frequency of code generations that appear to match code snippets in the training data'[98]. Furthermore, they also claim that where code was reproduced verbatim, these 'rare instances' comprised 'common expressions or conventions within the programming language that appeared over and over again in the training data'[99]. They also suggested that these anomalies were due to the 'predictive weightings in the model rather than retention and copying of specific code'. Essentially, they state that there was no actual memorisation that would lead to verbatim reproductions of code, but rather that the resulting code is statistically very likely to occur, given the fact that it is probably the most efficient way to express a particular functionality in the respective programming language. Finally, they seemed to suggest that the responsibility for the generated code lies with the user anyway as it is a customised response to the user's prompt.[100]

However, these claims hardly stand the test of the following example of code recited verbatim by an early version of GitHub Copilot:

```
float Q_rsqrt( float number )
{
        long i;
        float x2, y;
        const float threehalfs = 1.5F;

        x2 = number * 0.5F;
        y  = number;
        i  = * ( long * ) &y;                  // evil floating point bit level
hacking
        i  = 0x5f3759df - ( i >> 1 );                // what the fuck?
        y  = * ( float * ) &i;
        y  = y * ( threehalfs - ( x2 * y * y ) );    // 1st iteration
//      y  = y * ( threehalfs - ( x2 * y * y ) );    // 2nd iteration, this can
be removed

        return y;
}
```

FIGURE 7 EXAMPLE OF REPRODUCED SOURCE CODE FROM QUAKE III

This snippet reproduces Greg Walsh's implementation of the fast inverse square root method from Quake III, alongside the author's original comments[101]. The fast inverse square root algorithm and Walsh's implementation have proven pivotal in video game graphics. The code snippet is particularly original because it implements a single Newton-Raphson iteration starting with a clever first guess. It is therefore not a functional piece of code but arguably

[98] Chen and others (n 23) 13. Citing Albert Ziegler, 'GitHub Copilot Research Recitation' (*The GitHub Blog*, 30 June 2021) <https://github.blog/2021-06-30-github-copilot-research-recitation/> accessed 2 August 2022.
[99] Chen and others (n 23) 13.
[100] ibid.
[101] 'Fast Inverse Square Root', , *Wikipedia* (2022) <https://en.wikipedia.org/w/index.php?title=Fast_inverse_square_root&oldid=1127370312> accessed 19 December 2022. Also Rys, 'Beyond3D - Origin of Quake3's Fast InvSqrt() - Part Two' (*Beyond3D*, 19 December 2006) <https://www.beyond3d.com/content/articles/15/> accessed 19 December 2022.

an original expression of an author's own intellectual creation. [102] On top of that, it even reports the author's comment, something that has no bearing in a functional code analysis and can only be explained only with verbatim copying. The distinction between functional and non-functional code is particularly important in the context of copyright analysis of computer programs. Functional code would typically not garner protection because it is not an original expression or because it can be expressed in a very limited number of ways. While the code snipped in the example above was originally released under the GPLv2, Copilot's autocompletion did not reproduce the licence text or provide attribution, sparking controversy in free and open source communities.

In a recent self-reported experiment, GitHub's Albert Ziegler tried to quantify Copilot's recitation capabilities. The overlap investigation was limited to 60 'words' - the quantitative criterion used is a subjectively chosen number. Excluded from the investigation were (1) repetitive suggestions close to what a programmer has already typed; (2) repetitive sequences, e.g., blocks of '<p>'; (3) standard inventories, e.g., prime numbers, the Greek alphabet etc.; and (4) common or even universal ways of doing something, e.g., using the BeautifulSoup package to parse a Wikipedia list, similar to matter-of-fact statements in natural language and not quotes. One of the outcomes of the experiment is that Copilot quotes when it lacks specific context and when the snippet exists in many different files in the training data. For example, it suggested starting an empty file with something it had even seen more than 700,000 different times during training – the text of the GNU General Public License! Furthermore, it seems that Copilot quotes mostly in generic contexts, i.e., suggestions at the beginning could be hit-and-miss.

In his conclusions, Ziegler recommended the development of user-facing tools to inform programmers when snippets are copied from the training data set so that they can decide whether to provide attribution. Indeed, GitHub Copilot's team has recently introduced a filter with the ability to identify strings matching public code with a reference to those repositories[103]. According to the FAQ section in GitHub Copilot's documentation:

> ***"With the filter enabled, GitHub Copilot checks code suggestions with its surrounding code for matches or near matches (ignoring whitespace) against public code on GitHub of about 150 characters. If there is a match, the suggestion will not be shown to you. In addition, we have announced that we are building a feature that will provide a reference for suggestions that resemble public code on GitHub so that you can make a more informed decision about whether and how to use that code, as well as explore and learn how that code is used in other projects."***[104]

Interestingly, this seems to suggest that GitHub Copilot's developers quantify the threshold of potential copyright infringement at about 150 characters, which is, roughly, between 20 and 40 words. While courts have attempted to apply (somewhat) quantitative approaches in their own infringement analysis[105], these are not based on any explicit metrics. Any such estimate on the part of a company is, therefore, arbitrary at best.

Codex is not a publicly available model, so we cannot inspect it. It is therefore impossible to say if it has memorised portions of its training data, whether for lack of generalisation or because of the data's long-tailed distribution. However, based on the empirical observations reported in the previous paragraphs, we can conclude that it has certainly memorised at least some of its training data and that at least some of them can qualify as expressive computer programs that are not purely functional and constitute works protected by copyright. This implies that if memorisation

[102] Chris Lomont, 'Fast Inverse Square Root'.
[103] Ryan J Salva, 'Introducing Code Referencing for GitHub Copilot' (*The GitHub Blog*, 3 August 2023) <https://github.blog/2023-08-03-introducing-code-referencing-for-github-copilot/> accessed 26 October 2023. See also Salva (n 58).
[104] See GitHub (n 3).
[105] *Infopaq I & II* (*11* words), *Football Association Premier League* ('*fragments*' within memory), *Public Relations Consultants Association* (*copies* on screen and in cache), and *SAS Institute* (third party procuring *part of* the source code to create, with the aid of that code, similar elements in a new program).

presents a tangible concern for users of language models, mitigation measures should be put in place both prior to and after releasing a model to the public.

5. Mitigating memorisation: sanitisation, differential privacy, data curation or absolution?

As previously said, much of the debate on memorisation has taken place in the context of machine learning privacy. Therefore, mitigation measures have similarly focused on preventing models from leaking personal data. In this section, we discuss these approaches and assess their usefulness for identifying and preventing copyright infringement. We also offer three possible solutions to the problem of memorisation from a copyright law perspective.

It has been argued that (privacy-preserving) language models rely on approaches based on wrong assumptions about natural language data.[106] The two prevalent approaches concern removing private information through data sanitisation and designing algorithms that do not memorise private training data by satisfying differential privacy.

5.1. Data sanitisation

Data sanitisation relies on the assumption that it is possible to formalise the concept of 'personal data' and, consequently, to design efficient algorithms to identify and remove this data according to a specification.[107] These assumptions have been challenged on three grounds. First, personal data is context-dependent[108] and, as explained, models do not understand context. Second, personal data rarely has clearly defined boundaries. Virtually any piece of information may become personal data depending on the context.[109] Finally, personal data is not a discrete unit as changes in context may often determine whether a piece of information is personal data or not. This is largely ignored by sanitisation algorithms which assume that information is discrete and, therefore, the determination of whether a particular data item is a binary variable.[110]

In our view, similar arguments can be made about the applicability of data sanitisation algorithms to prevent copyright infringement.

First, computer programs are hybrid artifacts with both expressive and functional elements. Copyright law protects only the expression of a program, but not the underlying ideas. The assessment is based on the qualitative criterion of originality, that is, whether a program constitutes an author's own intellectual creation. Under EU law, it factors in considerations such as whether the work reflects the author's personality and expresses their free and creative choices in the production of the work. Deciding whether a computer program is purely functional or expressive is therefore not a matter of assigning a value to a binary variable.

Second, much like identifiability in data protection law, originality is not a discrete unit of copyright law. Courts have been more willing to view computer programs as tools rather than as expressive works. As discussed above, the CJEU has adopted a more nuanced approach arguing that 'neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and, as such, are not protected by copyright in computer programs…'[111]. However, the CJEU has also stated that '… if a third party were to procure the part of the source code or the object code relating to the programming language or to the format of data files used in a computer program, and

[106] Brown and others (n 9) 1.
[107] ibid 10.
[108] Nadezhda Purtova, 'The Law of Everything. Broad Concept of Personal Data and Future of EU Data Protection Law' (2018) 10 Law, Innovation and Technology 40, 44.
[109] ibid 57–59. Also Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) 2016 (OJ L), recital 26.
[110] Brown and others (n 9) 10–11.
[111] *SAS Institute Inc v World Programming Ltd* [2012] ECLI:EU:C:2012:259 (Court of Justice of the European Union) [39–46].

if that party were to create, with the aid of that code, similar elements in its own computer program, that conduct would be liable to constitute partial reproduction…'.[112] Whether a computer program is expressive or functional is, therefore, an evaluation to be made along a continuum rather than by assigning a discrete value.

Finally, applying data sanitisation algorithms at scale would arguably be too challenging for data originating from publicly available web sources. The sheer amount of data potentially covered by copyright would require considerable efforts on the part of model developers to simply sift through. Furthermore, as explained, data sanitisation entails qualitative assessments as to when a particular training example may or may not be covered by copyright law. In the context of works such as computer programs this assessment is even more challenging as it involves distinguishing between functional and expressive elements.

5.2. Differential privacy

Designing algorithms that do not memorise private training data by satisfying differential privacy (DP) is another approach to addressing the data protection concerns with language models. Could this approach be beneficial for preventing copyright infringement too?

Differential privacy can be defined as a 'data protection measure designed to assure users that contributing their data to a dataset will not reveal much additional information about the user, when the result of a DP algorithm trained on the dataset is released'[113]. However, satisfying differential privacy requires a unified definition of private data. [114] This is challenging for language data because of the communicative nature of language. Even if we remove all messages about one particular user, this may prove insufficient to remove the private information from the training set because other records might reference that same information.[115] Furthermore, to protect a specific unit of data is not tantamount to satisfying legal obligations of data protection. As discussed, personal data is contextual and not necessarily limited to pieces of information originating only from the data subject. One piece of information can easily span across data concerning more than one individual. Therefore, the differential privacy guarantees that a model would not leak individual records from the training data set cannot satisfy the legal data protection requirements.

Similarly, designing algorithms that satisfy differential privacy are unlikely to be of much use in preventing copyright infringement. Crucially, differential privacy does not prevent memorisation that may occur across a large data distribution. This is 'particularly problematic for copyrighted works, which might appear thousands of times across the Web'[116]. Even if differential privacy can prevent a model from *leaking* memorised data, it does prevent *memorisation* in the first place. While this is certainly helpful for the model's end users, it is not a solution to the fundamental issue of memorisation.

5.3. Data curation

One way to mitigate memorisation is to acknowledge that training models with bulk data from the Web is fundamentally unsound and that we need to invest more in the development of curated datasets and making the procurement of such datasets easier. In the EU, legal instruments such as the Open Data Directive and the AI Act proposal have explicitly identified the need for high-quality datasets for AI training.[117]

The problem with using training data in bulk from the Web has two facets. One concerns the lack of interpretability of non-deterministic black-box models with hidden internal states, such as the majority of commercially available language models. Interpretability should be an integrated component of models that can exhibit potentially infringing behaviour. Another concern is the need for data sanitisation which, as shown, cannot be left to algorithmic solutions alone.

[112] ibid.
[113] Brown and others (n 9) 11.
[114] ibid 11–12.
[115] ibid 12.
[116] Wallace and others (n 56).
[117] See e.g., Rec. 9 Directive (EU) 2019/1024 of the European Parliament and of the Council of 20 June 2019 on open data and the re-use of public sector information (recast).

There is mounting evidence that the wholesale use of data scraped from the Web is a bad practice in general. The phenomenon of memorisation is yet another reason to rethink our model training approaches. It has long been argued that 'if even a small fraction of the millions of dollars that are invested into training language models were instead put into collecting better training data, significant progress could be made to mitigate language models' harmful side effects'[118].

New data curation approaches can take very many different shapes. For example, the open source development model may inspire efforts to create community annotated data sets for specific domains. Furthermore, businesses that specialise in data annotation already provide access to vetted, high-quality data sets for different domains. Additionally, companies that develop large upstream models that are likely to be fine-tuned for different downstream tasks should contribute to the development of best practices for data curation.

Finally, users of both curated data sets and resulting models would require legal certainty in the form of immunity to copyright infringement claims. For example, Microsoft has recently announced its Copilot Copyright Commitment which is, essentially, a copyright indemnification pledge. In the words of Microsoft's President Brad Smith and Chief Legal Officer Hossein Nowbar, 'if a third party sues a commercial customer for copyright infringement for using Microsoft's Copilots or the output they generate, we will defend the customer and pay the amount of any adverse judgments or settlements that result from the lawsuit, as long as the customer used the guardrails and content filters we have built into our products'.[119] A full analysis of the legal effects of this pledge is beyond the scope of this article. Nevertheless, this type of indemnification commitments could place already dominant platforms and software companies in a much more favourable position vis-à-vis business customers, compared to companies offering so-called 'open source' models which cannot offer similar indemnities.

5.4. Absolution

Our last proposal is perhaps also the most radical as it suggests absolving model builders from liability for copyright infringement by accepting the view that memorisation is part of an essentially technical process. The (permanent) 'copy' created in the learning process does not have an independent value and is fundamentally different from a copy in the traditional sense understood in copyright law.

As discussed in the beginning of this article, the use of training data in machine learning entails a process of deconstruction of the input data as tokens. For text, a token can be a word, part of a word (subword), or indeed a character. These tokens are denoted numerically as semantically useful units in the form of vectors which represent the meaning of a token relative to other tokens. Essentially, this process involves transformation of copyright protected works, for example, a short story, into numerical values. Does copyright extend to this 'transformed' subject matter? During tokenisation, the work is literally deconstructed into building blocks by splitting words, phrases, sentences, and paragraphs. If we merely extract the meaning of and relations between these microscopic building blocks of the language that express the work in an objective form, then this can hardly be seen as reproduction of the work. The numerical representation of (parts of) a work into a collection of semantically useful units is also not an adaptation of the work either because the right of adaptation anticipates the creation of a new work based on the original work. Tokens are mapped to vectors, which are then passed into neural networks, so copyright eligible subject matter is nowhere to be found in the process, i.e., there is no new 'work'. In light of this analysis, it is arguable that the extracted information deconstructed as tokens and represented numerically falls completely outside the remit of copyright. The protected work becomes so 'decomposed' that it can no longer be reconstructed back or perceived as a 'work' and is therefore an independent object detached from the original protected work. In other words, numerical data are not copyrightable.

One possible objection to this conclusion is that for computer programs copyright law protects both source code and object code, which is, in a way, a numerical representation of the source code. The main argument would be that the object code is just another representation of the source code and, as such, the two should be treated as one 'work'.

[118] Wallace and others (n 56).

[119] Brad Smith and Hossein Nowbar, 'Microsoft Announces New Copilot Copyright Commitment for Customers' (*Microsoft On the Issues*, 7 September 2023) <https://blogs.microsoft.com/on-the-issues/2023/09/07/copilot-copyright-commitment-ai-legal-concerns/> accessed 26 October 2023.

This is a clearly established and uncontested position in both law and practice. However, we argue that the case with the representation of a work as tokens and vectors is different from compiled object code for the following reasons:

- Object code is an executable form of source code that has been optimised by the compiler and linker in a way that allows it to run on a particular computer architecture. The tokenisation of a sentence merely deconstructs the building elements of the language of that sentence into smaller tokens. The numerical representation of these tokens as vectors reflects the meaning of a token relative to other tokens but is not itself a different form of the work.
- Object code is a computational artefact that represents the whole work. It is the complete translation of a program expressed in source code into an executable form. In contrast, the numerical representation of a part of, say, a literary work, provides information about the structural elements of the work's language and structure, but it tells us nothing about its aesthetic or artistic merits. To the untrained eye, reading this numerical representation would be 'as useful as drinking a glass of diesel oil instead of pouring it into the tank of his vehicle', to borrow from the Advocate General in the *Tom Kabinet* case.[120]
- Object code represents the complete translation of source code into an executable binary. In contrast, the vector representation of tokens relative to other tokens is merely an intermediary step in a long technical process. It does not represent the complete work and the complete work is not used 'as such' in the training process.

Absolution from copyright liability should not be seen as a *carte blanche* for model developers to simply scrape at scale whatever data may be publicly available on the web, which is what most have been doing until now anyway. Any such exemption from liability should come with a proof of accountability on the part of developers. This may include the application of mandatory use of state-of-the-art metrics (e.g., exposure metric that uses an individual canary[121]) to measure memorisation and a corresponding catalogue of mitigation measures. Some of these measures may be in a less technical form already recognised in the latest discussions on the AI Act proposal. Accountability could also link public interest elements that attach to a "generative AI copyright exception", such as disclosure of certain details about the training data.

Finally, we have the example of safe harbours in the EU E-Commerce Directive. A similar legal device for generative AI producers could prove beneficial in supporting initial investments in AI technology. After an initial period of, say, 10 or 20 years, the legislator could change the liability profiles of these actors, much like it did with the Digital Services Act or Article 17 CDSM Directive. The principle is the same: the difference is in the pace. For example, 30 years ago, when the first hosting providers appeared, it was not clear what technological social and economic changes they would cause. The first version of YouTube is still shrouded in mystery.

5.5. FINAL REMARKS

The social, labour and economic impact of generative AI is vast. They require solutions that lie far beyond the purview of copyright. Of course, copyright could be part of the solution, however the use of copyright as the principal lever for AI regulation appears imperfect and incomplete. Possible solutions from a copyright point of view include safe harbour provisions that oblige providers to declare that a particular artifact is machine-generated, the inclusion of watermarking, disclosure of training material in a safe environment etc. This approach would effectively exonerate or mitigate copyright infringement, or anyway prevent the seeking of damages, with the only possibility of injunctive

[120] *Opinion of Advocate General Szpunar delivered on 10 September 2019 Nederlands Uitgeversverbond and Groep Algemene Uitgevers v Tom Kabinet Internet BV and Others Request for a preliminary ruling from the Rechtbank Den Haag Reference for a preliminary ruling — Harmonisation of certain aspects of copyright and related rights in the information society — Directive 2001/29/EC — Article 3(1) — Right of communication to the public — Making available — Article 4 — Distribution right — Exhaustion — Electronic books (e-books) — Virtual market for 'second-hand' e-books Case C-263/18* (ECJ) [57].
[121] Carlini and others, 'The Secret Sharer' (n 39) 5.

relief, such as an injunction to remove data from the model,[122] or indeed model disgorgement[123] as a measure of last resort.

ACKNOWLEDGMENTS

This research is funded by the European Union's Horizon Europe research and innovation programme under the 3Os and IP awareness raising for collaborative ecosystems (ZOOOM) project, grant agreement No 101070077. The authors have contributed equally to this work.

REFERENCES

Brown G and others, ‘When Is Memorization of Irrelevant Training Data Necessary for High-Accuracy Learning?’, *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (2021) <http://arxiv.org/abs/2012.06421> accessed 8 November 2022

Brown H and others, ‘What Does It Mean for a Language Model to Preserve Privacy?’ (arXiv, 14 February 2022) <http://arxiv.org/abs/2202.05520> accessed 8 November 2022

Butterick M and others, ‘GitHub Copilot Litigation’ (*GitHub Copilot litigation*, 3 November 2022) <https://githubcopilotlitigation.com/> accessed 13 March 2023

Carlini N and others, ‘The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks’ (arXiv, 16 July 2019) <http://arxiv.org/abs/1802.08232> accessed 28 July 2022

——, ‘Extracting Training Data from Large Language Models’ (2021) <https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting> accessed 8 November 2022

——, ‘Quantifying Memorization Across Neural Language Models’ (arXiv, 15 February 2022) <http://arxiv.org/abs/2202.07646> accessed 28 July 2022

——, ‘The Privacy Onion Effect: Memorization Is Relative’ (arXiv, 22 June 2022) <http://arxiv.org/abs/2206.10469> accessed 8 November 2022

Chen M and others, ‘Evaluating Large Language Models Trained on Code’ [2021] arXiv:2107.03374 [cs] <http://arxiv.org/abs/2107.03374> accessed 25 November 2021

Cheng C, Duchi J and Kuditipudi R, ‘Memorize to Generalize: On the Necessity of Interpolation in High Dimensional Linear Regression’ (arXiv, 16 June 2022) <http://arxiv.org/abs/2202.09889> accessed 10 May 2023

‘Content Mining - Creative Commons’ <https://wiki.creativecommons.org/wiki/Content_mining> accessed 15 May 2023

Ducato R and Strowel AM, ‘Ensuring Text and Data Mining: Remaining Issues With the EU Copyright Exceptions and Possible Ways Out’ (1 February 2021) <https://papers.ssrn.com/abstract=3829858> accessed 26 October 2023

Dugas D, ‘The GPT-3 Architecture, on a Napkin’ (*Robotics & Machine Learning*, 2022) <https://dugas.ch/artificial_curiosity/GPT_architecture.html> accessed 10 May 2023

[122] This is, however, highly unlikely to work in practice because of the way models are built. The high dimensionality of these models means that not even their creators can modify or remove individual data points.
[123] Joshua A Goland, ‘Algorithmic Disgorgement: Destruction of Artificial Intelligence Models as the FTC’s Newest Enforcement Tool for Bad Data’ (1 March 2023) <https://papers.ssrn.com/abstract=4382254> accessed 26 October 2023.

‘Fast Inverse Square Root’, , *Wikipedia* (2022) <https://en.wikipedia.org/w/index.php?title=Fast_inverse_square_root&oldid=1127370312> accessed 19 December 2022

Feldman V, ‘Does Learning Require Memorization? A Short Tale about a Long Tail’, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery 2020) <http://doi.org/10.1145/3357713.3384290> accessed 8 November 2022

Feldman V and Zhang C, ‘What Neural Networks Memorize and Why: Discovering the Long Tail via Influence Estimation’ (arXiv, 9 August 2020) <http://arxiv.org/abs/2008.03703> accessed 18 November 2022

Fiil-Flynn SM and others, ‘Legal Reform to Enhance Global Text and Data Mining Research’ (2022) 378 Science 951

GitHub, ‘GitHub Copilot - Your AI Pair Programmer - Frequently Asked Questions’ (*GitHub*, 2023) <https://github.com/features/copilot> accessed 19 December 2022

Goland JA, ‘Algorithmic Disgorgement: Destruction of Artificial Intelligence Models as the FTC’s Newest Enforcement Tool for Bad Data’ (1 March 2023) <https://papers.ssrn.com/abstract=4382254> accessed 26 October 2023

Jurafsky D and Martin JH, *Speech and Language Processing* (Third Edition draft, 2022)

Kuhn B, ‘If Software Is My Copilot, Who Programmed My Software?’, *Free Software Foundation* (2022)

Lomont C, ‘Fast Inverse Square Root’

Margoni T and Kretschmer M, ‘A Deeper Look into the EU Text and Data Mining Exceptions: Harmonisation, Data Ownership, and the Future of Technology’ (Zenodo 2021) <https://zenodo.org/record/5082012> accessed 6 July 2022

Pendant, ‘Master List of Lawsuits v. AI, ChatGPT, OpenAI, Microsoft, Meta, Midjourney & Other AI Cos.’ (*Chat GPT Is Eating the World*, 19 October 2023) <https://chatgptiseatingtheworld.com/2023/10/19/master-list-of-lawsuits-v-ai-chatgpt-openai-microsoft-meta-midjourney-other-ai-cos/> accessed 26 October 2023

Purtova N, ‘The Law of Everything. Broad Concept of Personal Data and Future of EU Data Protection Law’ (2018) 10 Law, Innovation and Technology 40

Rys, ‘Beyond3D - Origin of Quake3’s Fast InvSqrt() - Part Two’ (*Beyond3D*, 19 December 2006) <https://www.beyond3d.com/content/articles/15/> accessed 19 December 2022

Salva RJ, ‘Preview: Referencing Public Code in GitHub Copilot’ (*The GitHub Blog*, 1 November 2022) <https://github.blog/2022-11-01-preview-referencing-public-code-in-github-copilot/> accessed 15 May 2023

——, ‘Introducing Code Referencing for GitHub Copilot’ (*The GitHub Blog*, 3 August 2023) <https://github.blog/2023-08-03-introducing-code-referencing-for-github-copilot/> accessed 26 October 2023

Smith B and Nowbar H, ‘Microsoft Announces New Copilot Copyright Commitment for Customers’ (*Microsoft On the Issues*, 7 September 2023) <https://blogs.microsoft.com/on-the-issues/2023/09/07/copilot-copyright-commitment-ai-legal-concerns/> accessed 26 October 2023

Sobel BLW, ‘Artificial Intelligence’s Fair Use Crisis’ (2018) 41 The Columbia Journal of Law & the Arts 45

Solar-Lezama A, ‘LLMs for Code’ (Generative AI: Language, Images and Code - a conversation with CSAIL 2023, Cambridge, MA, USA, 10 April 2023) <https://www.youtube.com/watch?v=VDKkc36lGeg> accessed 24 October 2023

Stefan Karpinski [@StefanKarpinski], 'In Case It's Not Clear What's Happening Here: @github's Copilot "Autocompletes" the Fast Inverse Square Root Implementation from Quake III — Which Is GPL2+ Code. It Then Autocompletes a BSD2 License Comment (with the Wrong Copyright Holder). This Is Fine.' <https://twitter.com/StefanKarpinski/status/1410971061181681674> accessed 26 October 2023

Vaswani A and others, 'Attention Is All You Need' (arXiv, 5 December 2017) <http://arxiv.org/abs/1706.03762> accessed 15 November 2022

Veale M, Binns R and Edwards L, 'Algorithms That Remember: Model Inversion Attacks and Data Protection Law' (2018) 376 Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 20180083

Wallace E and others, 'Does GPT-2 Know Your Phone Number?' (*The Berkeley Artificial Intelligence Research Blog*, 20 December 2020) <http://bair.berkeley.edu/blog/2020/12/20/lmmem/> accessed 19 December 2022

Wang K and others, 'Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 Small' (arXiv, 1 November 2022) <http://arxiv.org/abs/2211.00593> accessed 7 February 2024

Zheng X and Jiang J, 'An Empirical Study of Memorization in NLP' [2022] Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 2022 May 22 - 27 6265

Zhu X, Anguelov D and Ramanan D, 'Capturing Long-Tail Distributions of Object Subcategories', *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014) <https://ieeexplore.ieee.org/document/6909517> accessed 24 October 2023

Ziegler A, 'GitHub Copilot Research Recitation' (*The GitHub Blog*, 30 June 2021) <https://github.blog/2021-06-30-github-copilot-research-recitation/> accessed 2 August 2022

Authors Guild v Google, Inc (2014) 804 F 3d 202 (Court of Appeals, 2nd Circuit)

Football Association Premier League and Others [2011] Court of Justice of the European Union C-403/08, ECLI:EU:C:2011:631

Infopaq International A/S v Danske Dagblades Forening [2009] ECJ Case C-5/08, ECLI:EU:C:2009:465

Maryland v Baltimore Radio Show, Inc, (1950) 338 338 US 912 912 (US Supreme Court)

Opinion of Advocate General Szpunar delivered on 10 September 2019 Nederlands Uitgeversverbond and Groep Algemene Uitgevers v Tom Kabinet Internet BV and Others Request for a preliminary ruling from the Rechtbank Den Haag Reference for a preliminary ruling — Harmonisation of certain aspects of copyright and related rights in the information society — Directive 2001/29/EC — Article 3(1) — Right of communication to the public — Making available — Article 4 — Distribution right — Exhaustion — Electronic books (e-books) — Virtual market for 'second-hand' e-books Case C-263/18 (ECJ)

Pelham GmbH, Moses Pelham, Martin Haas v Ralf Hütter, Florian Schneider-Esleben [2019] Court of Justice of the European Union C-476/17, ECLI:EU:C:2019:624

SAS Institute Inc v World Programming Ltd [2012] ECLI:EU:C:2012:259 (Court of Justice of the European Union)

Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases 1996 (OJ L)

Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society 2001

Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs (Codified version) 2009

Directive (EU) 2019/790 of the European Parliament and of the Council of 17 April 2019 on copyright and related rights in the Digital Single Market and amending Directives 96/9/EC and 2001/29/EC 2019 (OJ L)

Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA relevance) 2016 (OJ L)