Self-Supervised Pretraining auf CIFAR-10 mit Masked Autoencodern

---

Patrick Radau

12. August 2025

Johannes Gutenberg University Mainz
FB08
Institute of Computer Science
Visual Computing

Bachelorarbeit

Self-Supervised Pretraining auf CIFAR-10 mit Masked Autoencodern

Patrick Radau

1. Gutachter Prof. Dr. Michael Wand
Institute of Computer Science - Visual Computing
Johannes Gutenberg University Mainz

2. Gutachter Prof. Dr. Elmar Schömer
Institute of Computer Science
Johannes Gutenberg University Mainz

Betreuer Prof. Dr. Michael Wand und
Prof. Dr. Elmar Schömer

12. August 2025

Patrick Radau
Self-Supervised Pretraining auf CIFAR-10 mit Masked Autoencodern
Bachelorarbeit, 12. August 2025
Gutachter: Prof. Dr. Michael Wand und Prof. Dr. Elmar Schömer
Betreuer: Prof. Dr. Michael Wand und Prof. Dr. Elmar Schömer

Johannes Gutenberg University Mainz
Visual Computing
Institute of Computer Science
FB08
Staudingerweg 9
55128 Mainz

Zusammenfassung

Das Problem vieler neuronaler Netze für die Bildverarbeitung, darunter auch Vision Transformer, ist, dass diese viele gut gelabelte Bilder benötigen. Große Mengen an Daten zu annotieren, ist nicht nur aufwendig, sondern auch noch kostenintensiv, gerade bei spezialisierten Inhalten mit begrenztem Datenzugang. Eine vielversprechende Lösung ist das Self-Supervised Pretraining, bei dem kaum noch gelabelte Daten nötig sind. Mit solch einer Lösung, dem sogenannten Masked Autoencoder, beschäftige ich mich in dieser Arbeit im Detail. Dieser bekommt maskierte Bilder übergeben und muss diese anhand des umliegenden Kontexts wieder zu einem ganzen Bild rekonstruieren. Durch diese Art von Pretraining erhält das Modell ein tiefes semantisches Wissen, welches dann für die spätere Klassifikation verwendet werden kann. Das Ziel dieser Arbeit ist es, eine Analyse zu erstellen, welche fundiert begründet, warum das Vortrainieren sinnvoll ist, aus welchen Gründen es funktioniert und vor allem, welche architektonischen Entscheidungen den Erfolg begünstigen. Dabei möchte ich eine Architektur für den kleinen CIFAR-10-Datensatz erstellen und diese optimieren.

In dieser Arbeit führe ich unterschiedlichste Experimente durch und evaluiere diese anhand von Benchmark-Ergebnissen auf ihren Erfolg in der Bildklassifikation. Dabei stelle ich zusätzlich Hypothesen für die Gründe der Ergebnisse auf.

Meine Ergebnisse zeigen, dass Masked Autoencoder (auch) für kleine Datensets funktionieren, aber womöglich nicht die beste Wahl für das Vortrainieren sind. Es hat sich gezeigt, dass das Rekonstruieren maskierter Bildbereiche zu einem tiefen semantischen Verständnis der zugrunde liegenden Bilder führt, was gerade für die Klassifikation von Bildern sinnvoll ist. Auch die Entscheidungen hinsichtlich der Architektur des Modells haben einen bedeutenden und signifikanten Einfluss auf die Trainingsqualität. Dennoch haben meine Ergebnisse auch gezeigt, dass der Masked Autoencoder sehr komplex ist und es leichtere Modelle gibt (SimMIM), um auf kleinen Datensets Self-Supervised Pretraining anzuwenden.

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Motivation . . . 1
- 1.2. Zielsetzung meiner Arbeit . . . 2
- 1.3. Aufbau der Arbeit . . . 2
- 1.4. Ergebnisse . . . 3

2. Grundlagen **5**
- 2.1. Supervised und Self-Supervised Learning . . . 5
- 2.2. Autoencoder: Struktur und Funktionsweise . . . 6
- 2.3. Kontrastives Lernen . . . 8

3. Verwandte Arbeiten **11**
- 3.1. BERT . . . 11
- 3.2. Vision Transformer (ViT): Revolutionierung der Computer Vision 12
  - 3.2.1. Architektur und technische Innovation . . . 13
- 3.3. Historie Masked Image Modeling . . . 13
- 3.4. Masked Autoencoder Are Scalable Vision Learners . . . 15
- 3.5. Weitere Forschungsarbeiten . . . 16

4. Masked Autoencoder **17**
- 4.1. Ansatz zum Masked Autoencoder . . . 17
- 4.2. Architektur & Implementation von MAE . . . 18
- 4.3. Trainingsverlauf . . . 20
- 4.4. Klassifikation . . . 21

5. Untersuchungen **23**
- 5.1. Maskierungsrate . . . 23
  - 5.1.1. Signifikanztest . . . 24
- 5.2. Maskierungsart . . . 29
  - 5.2.1. Ergebnisse & Interpretation . . . 29
  - 5.2.2. Fazit . . . 31
- 5.3. Patch-Größe . . . 31
  - 5.3.1. Ergebnisse & Interpretation . . . 32

5.4. Decoder 32
5.4.1. Decoder Tiefe 33
5.4.2. Embedding-Dimension 34
5.4.3. Dynamische Token-Interaktion 35
5.4.4. Fazit 37
5.5. Datenaugmentation 37
5.5.1. Ergebnisse & Interpretation 38
5.5.2. Fazit 38
5.6. Loss Funktion 39
5.6.1. Ergebnisse & Interpretation 40
5.6.2. Fazit 41

6. Kritische Hinterfragung meiner Untersuchung 43

7. Vertiefung der Funktionsweise 45
7.1. Rolle der Maskierungsrate 45
7.1.1. Untersuchung / Experiment 47
7.2. Verbindung zu kontrastivem Lernen 48
7.2.1. Motivation 49
7.2.2. Intuitive Erklärung 49
7.2.3. Mathematische Verbindung 50
7.2.4. Grund für die hohe Maskierungsrate 52
7.2.5. Zusammenfassung 53

8. Fazit Masked Autoencoder 55
8.1. Schlussendliches Ergebnis 55
8.2. Schlussfolgerung 56

9. SimMIM 57
9.1. Vorgehensweise und Rekonstruktion 57
9.2. Ergebnisse und Interpretation 58

10. Zusammenfassung und Ausblick 61

Literatur 63

Abbildungsverzeichnis 69

Tabellenverzeichnis 71

Programmausdrucksverzeichnis 73

A. Appendix **75**
A.1. Unterschiedliche Maskierungsstrategien und deren Rekonstruktion 76
A.2. Ablauf aller Untersuchungen 77
A.3. Weitere nicht aufgelistete Untersuchungen 78
A.3.1. ImageNet Pretraining 78
A.3.2. CIFAR-100 78

1 Einleitung

1.1 Motivation

Viele Supervised Learning Verfahren haben bahnbrechende Erfolge erzielt, jedoch stellt die manuelle Beschaffung von Daten und die Annotierung mit Labels eine erhebliche finanzielle und auch zeitliche Herausforderung dar [24], darüber hinaus werden meist Personen aus dem globalen Süden für diese Arbeit ausgebeutet [@22, @11].

Deshalb hat sich seit mehreren Jahren in Sprachmodellen Self-Supervised Learning als Paradigmenwechsel etabliert [23, Kapitel 10]: Hierbei werden Modelle trainiert, welche strukturelle Muster in ungelabelten Textdaten erkennen sollen. Das von Google vorgestellte Sprachmodell BERT (2019) [14] (auf BERT wird in Kapitel 3.1 nochmal genauer eingegangen) hat einen Durchbruch im Bereich der natürlichen Sprachverarbeitung dargestellt und ist eine Art des Self-Supervised Learnings. Durch das Maskieren von Wörtern innerhalb eines Satzes und das anschließende Vorhersagen der Wörter, basierend auf dem umgebenden Kontext, entsteht ein tieferes sprachliches Verständnis ganz ohne gelabelte Daten [14, 37, @36].

Diese Idee des Maskierens und Vorhersagens hat nicht nur die Sprachverarbeitung revolutioniert, sondern auch den Bereich der Bildverarbeitung. Mit der Einführung von Vision Transformern (ViT) [15], welche die gegebenen Bilder in einzelne kleine Patches unterteilen, wurde es möglich, Self-Supervised Learning auch effektiv und mit sehr guter Performance auf visuelle Inhalte zu übertragen. Dieselbe Funktionsweise wie bei BERT funktioniert nun bei Bildern, man kann den ViT mit einem Masked Autoencoder (MAE) [17] vortrainieren, indem Teile des Bildes maskiert werden. Dabei entsteht ein tiefes, semantisches Bildverständnis, ganz ohne gelabelte Daten. Das Modell erschließt sich die Strukturen und Formen des Bildes alleine. [17]

Das self-supervised trainierte Netzwerk kann anschließend mit gelabelten Daten (also supervised) feinjustiert werden, um die spätere Klassifikation oder andere Downstream-Tasks erfolgreich auszuführen.

Mit dem Hintergrund steigender Anforderungen an die Skalierbarkeit und dem Fehlen von qualitativ gut annotierten Daten gewinnt diese Methode enorm an Bedeutung. Mit dem zusätzlichen tiefen Verständnis des Bildes gelangen hervorragende Ergebnisse in vielen Downstream-Tasks [17]. Die aktuellen Forschungen beziehen sich jedoch alle auf große Datensets wie ImageNet [13] und nicht auf Datensets, welche wenige Zehntausende Bilder haben [17, 46, 15]. In manchen Bereichen ist es jedoch einfach nicht möglich, mehrere Hunderttausend bis Millionen Bilder zu erstellen. Aus diesen Gründen will ich mich in dieser Arbeit tiefer in das Thema einarbeiten und Masked Autoencoder im Bereich der Bildklassifikation untersuchen und verstehen. Dabei gehe ich auf kleine Datensets ein, genauer gesagt auf den CIFAR-10-Datensatz [26].

1.2 Zielsetzung meiner Arbeit

Ziel meiner Arbeit ist es, die Arbeit von Kaiming He et al. (2021) mit dem Titel "Masked Autoencoders Are Scalable Vision Learners" [17] nachzuvollziehen und eine passende Autoencoder-Struktur für das CIFAR-10-Datenset sowie für weitere kleine Datensets zu konstruieren. Dabei möchte ich auch auf weiterführende Forschungen eingehen, um meine Architektur weiter zu verbessern und die Funktionsweise zu erläutern. Ein weiterer Teil meiner Arbeit ist es, mithilfe von zwei unterschiedlichen Arbeiten [49, 25] herauszufinden, warum Masked Autoencoder funktionieren und welche Verbindungen es zu anderen Self-Supervised-Learning-Verfahren gibt.

Der Grund, die Untersuchungen anhand kleiner Datensets durchzuführen und hierfür ein passendes Modell zu finden, liegt unter anderem an der Tatsache, dass ich keine große Rechenleistung besitze. Allerdings kann ein Pretraining auf kleinem Datensatz, gerade wenn stark domänenspezifisches Wissen erlernt werden soll, auch Sinn ergeben und sollte deswegen untersucht werden.

1.3 Aufbau der Arbeit

Vor dem Verwandten-Arbeiten-Abschnitt 3 befasst sich meine Arbeit mit den Grundbegriffen, die für das Self-Supervised-Pretraining mit einem Mas-

ked Autoencoder nötig sind. Darunter zählt die Begriffserklärung von Self-Supervised-Learning und eine kurze Einführung zu Autoencodern sowie kontrastivem Lernen.

Weiter geht es mit der Einführung zu den Masked Autoencodern. In diesem Abschnitt wird die grundlegende Architektur, welche ich in meiner Arbeit verwende, erläutert. Aber auch der erste Trainingsverlauf und die ersten Rekonstruktionsergebnisse werden hier gezeigt.

Im nächsten Abschnitt 5 wird die grundlegende Architektur angepasst und auf mögliche Verbesserungen untersucht. Hier wird nicht nur auf das Paper von Kaiming He et al. eingegangen, sondern es werden auch eigene Ideen und Ideen aus verwandten und weiterführenden Papern [45] eingebracht. Hier wird unter anderem die Maskierungsrate, die Maskierungsart, der Decoder, die Data-Augmentation und die Loss-Funktion untersucht. Diese ganzen Untersuchungen werden dann noch einmal kritisch hinterfragt.

Anschließend gehe ich tiefer in die Mechanismen ein, welche für den Erfolg des Masked Autoencoders ausschlaggebend sind. Ich erkläre, welche Rolle die Maskierungsrate in der Klassifikation spielt und welche Verbindung es zu verwandten Ansätzen des Self-Supervised Learnings gibt.

Nach einem kurzen Fazit zu Masked Autoencodern, werde ich die Komplexität des Modells hinterfragen und mit “A Simple Framework for Masked Image Modeling (SimMIM)” [44] ein ähnliches, aber leichtgewichtigeres Modell vorstellen und evaluieren, bevor ich zu meinem schlussendlichen Fazit der Arbeit komme.

1.4 Ergebnisse

Meine Ergebnisse zeigen, dass der Masked Autoencoder auch auf kleinen Datensets Verbesserungen von über 10 % ermöglicht (auf circa 90 %), jedoch nicht die einzige gute Wahl ist. Auch verwandte Netze wie SimMIM, welches ähnlich funktioniert, erreicht dieselben Ergebnisse, trotz vereinfachter Struktur. Beide Netze übertreffen Vanilla-ViT-Tiny-Modelle, jedoch nicht stark angepasste Vision-Transformer-Strukturen für CIFAR-10 [42, @10].

Beim Masked Autoencoder hat sich herausgestellt, dass die Maskierungsart und die Maskierungsrate einen großen Einfluss haben. Hier hat eine zufällige Maskierung mit einer Maskierungsrate von ungefähr 75-85 % am besten

abgeschnitten. Jedoch gibt es auch im Decoder Verbesserungen, wie die dynamische Tokeninteraktion, mit welcher leicht bessere Ergebnisse erzielt werden, oder eine vergrößerte Patch-Größe von 4x4. Ein vielleicht etwas verwunderndes Ergebnis auf den ersten Blick ist, dass die Daten-Augmentation keine Verbesserung bringt.

2 Grundlagen

2.1 Supervised und Self-Supervised Learning

Die rasante Entwicklung im Bereich des maschinellen Lernens bringt unterschiedliche Lernparadigmen hervor. Eines dieser Paradigmen ist das Self-Supervised Learning (SSL), welches sich in Hinblick auf die Nutzung und Annotation von Trainingsdaten gegenüber dem ursprünglichen Supervised Learning für die Bildklassifikation unterscheidet.

Supervised Learning: Das klassische Paradigma des maschinellen Lernens ist das Supervised Learning, bei dem Modelle anhand von gelabelten Daten trainiert werden. Das heißt, jeder Trainingspunkt besteht aus einem Eingabe-Ausgabe-Paar, die Eingabe stellt das Bild dar und die Ausgabe das zugehörige Label. Ziel ist es, durch das Trainieren des Netzes / Vision Transformer [15] auf neue Daten das passende Label vorherzusagen. Das heißt, es wird eine Funktion gelernt, welche auf gelabelten Daten trainiert wird. Ein wesentlicher Nachteil ist das schon angesprochene Annotieren der Daten, da dies sehr kosten- und zeitaufwendig ist [24]. [@43]

Self-Supervised Learning: Self-Supervised Learning geht das Problem des Supervised Learnings an, indem es Lernaufgaben definiert, bei denen das Modell die Labels automatisch aus den Rohdaten generieren muss und somit das Annotieren größtenteils entfällt [@4, 21]. Somit muss es nicht mehr händisch gemacht werden. Dabei lernt es, inhärente Strukturen und Zusammenhänge in den Daten zu erfassen. Diese Zusammenhänge lernt das Modell teilweise sogar stärker als mit dem Supervised-Learning-Ansatz [17]. Das Modell, welches durch Self-Supervised Learning trainiert wird, kann später mit wenigen annotierten Daten auf unterschiedliche Downstream-Tasks spezialisiert werden.

Im Kern zielt diese Methode darauf ab, im Pretraining einen Encoder (ein Modellteil, welches Eingabedaten in interne Repräsentationen umwandelt)

zu trainieren, statt eine einzelne Funktion, wobei der Lernprozess in zwei Phasen aufgeteilt ist:

1. Zunächst wird eine künstliche Lernaufgabe konstruiert, indem zu jedem Eingangsdatenpunkt x ein Zielwert t generiert wird. Dieses Ziel ist nicht vorgegeben, sondern das Modell leitet sich dieses Ziel aus den Daten selbst ab. [@4]

2. Anschließend wird mithilfe klassischer Supervised-Learning-Verfahren der Input x auf das Ziel t trainiert. [@4]

2.2 Autoencoder: Struktur und Funktionsweise

Der Autoencoder ist ein spezieller Typ künstlicher neuronaler Netze, der für das Self-Supervised Learning von komprimierten Datenrepräsentationen verwendet wird. Die Funktionsweise eines Autoencoders nach Umberto Michelucci [31] ist sehr einfach. Die grundlegende Idee ist, Eingabedaten (wie ein Bild, was ich hier als Beispiel auch weiter verwenden werde) in eine niedrig-dimensionale Repräsentation zu überführen, die dann wieder in das originale Bild rekonstruiert werden kann [31].

Er ist in drei unterschiedliche Bestandteile unterteilt:

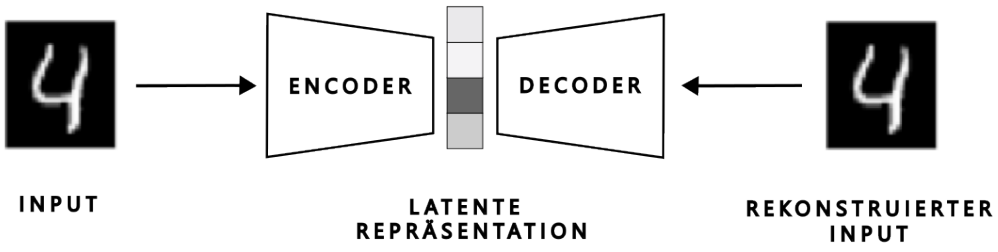


Abb. 2.1.: Struktur eines Autoencoders. Zu sehen ist der Input (Bild mit einer 4), welches in den Encoder übergeben wird. Der Encoder wandelt den Input in die latente Repräsentation um, welche in der Mitte als vier Rechtecke abgebildet ist. Der Decoder rekonstruiert nun das Bild wieder aus dem latenten Raum. Abbildung wurde leicht angepasst von Michelucci [31] übernommen

1. **Der Encoder.** Dies ist ein neuronales Netz, welches die Eingabe nimmt und diese komprimiert. Ziel des Encoders ist es, aus der hoch-dimensionalen Eingabe die relevantesten Merkmale zu extrahieren [31].

2. **Die latente Repräsentation.** Dies ist die niedrig-dimensionale Repräsentation des Bildes, welches nur noch die relevantesten Merkmale besitzt [31]. Das Bild befindet sich hier im sogenannten Bottleneck.

3. **Der Decoder.** Er versucht die latente Repräsentation der Eingabe wieder so gut wie möglich zu rekonstruieren [31]. Er besteht auch aus einem neuronalen Netz.

Das Ganze kann auch formal definiert werden. Der Encoder wird als Funktion $g(x_i)$ dargestellt, der Decoder als Funktion $f(h_i)$. Dabei ist x_i das Eingabebild und h_i die latente Repräsentation von x_i. Die latente Repräsentation lässt sich als Output h des Encoders verstehen $h = g(x_i)$ [31]. Des Weiteren wird die Eingabe auf die vom Encoder vorgesehene Feature-Dimensionalität q reduziert, also: $g : \mathbb{R}^n \rightarrow \mathbb{R}^q$. Der Decoder versucht dann, möglichst genau wieder das originale Bild herzustellen, was folgendermaßen aussieht [31]:

$$\tilde{x}_i = f(h_i) = f(g(x_i)).$$

Bei dem Trainieren des Autoencoders lernt der Encoder die wichtigsten q Merkmale aus der Eingabe herauszufiltern. Ein Merkmal kann zum Beispiel die Anzahl an Kanten sein. Der Decoder lernt beim Trainieren aus diesen Merkmalen wieder das ursprüngliche Bild herzustellen. Es muss also folgende Funktion [31] minimiert werden:

$$\arg\min_{f,g} < [\Delta(x_i, f(g(x_i)))] > .$$

Zusammenfassend lässt sich festhalten, dass Autoencoder ein effizientes Mittel zur Dimensionsreduktion und Merkmalsextraktion darstellen. Sie eignen sich insbesondere für Anwendungen, bei denen Daten komprimiert gespeichert oder analysiert werden sollen. Je nach Grad der Kompression (d.h. der Größe der Feature-Dimensionalität q) kann es jedoch zu großen Informationsverlusten und somit zu Abweichungen zwischen Original und Rekonstruktion kommen. [31]

2.3 Kontrastives Lernen

Kontrastives Lernen wird im späteren Verlauf meiner Arbeit eine Rolle spielen, deshalb eine kurze Einführung in dieses Teilgebiet der Self-Supervised Learning-Verfahren, welches ermöglicht, aussagekräftige Repräsentationen von Daten zu erlernen. Die Funktionsweise ist auf den ersten Blick leicht: ähnliche Datenpaare (positive pairs) werden in einen sogenannten Merkmalsraum nah aneinandergebracht und unähnliche Paare (negative pairs) sollen weit voneinander weg liegen [8]. Nun weiß das Modell am Anfang nicht, welche Bilder ähnlich sind oder wie sich diese Bilder voneinander unterscheiden, deshalb benötigen wir sogenannte initiale positive Paare [8, 46]. Diese werden meist durch eine Datenaugmentation erzeugt, welche gleich als Teil der Architektur nochmals erläutert wird. Durch diese initialen Paare lernt das Modell Bilder korrekt in dem Merkmalsraum zu legen. Im späteren Verlauf ist dann keine Augmentation mehr vonnöten.

Die Architektur eines kontrastiven Lernmodells besteht typischerweise aus vier Schlüsselkomponenten:

1. Die **Datenaugmentation** erzeugt verschiedene Ansichten desselben Bildes durch Transformationen. Diese unterschiedlichen Ansichten liegen im Merkmalsraum dann nahe beieinander und das Modell kann mit diesen anfangen zu lernen. [8]

2. Ein **Encoder**, welcher die augmentierten Bilder mithilfe eines neuronalen Netzes in eine latente Repräsentation abbildet. [8]

3. Ein **Projektionskopf**, welcher auf der latenten Repräsentation des Encoders angewendet wird. Dieser Schritt projiziert die Repräsentationen in einen Merkmalsraum, in dem der kontrastive Lernprozess dann stattfinden kann. [8]

4. Die **kontrastive Verlustfunktion**, erzeugt dann einen Merkmalsraum, welcher für unterschiedlichste Aufgaben, wie auch die Klassifikation geeignet ist. Dies passiert indem die Distanz zu negativen Paaren maximiert und die zu positiven Paaren optimiert / minimiert wird. [8]

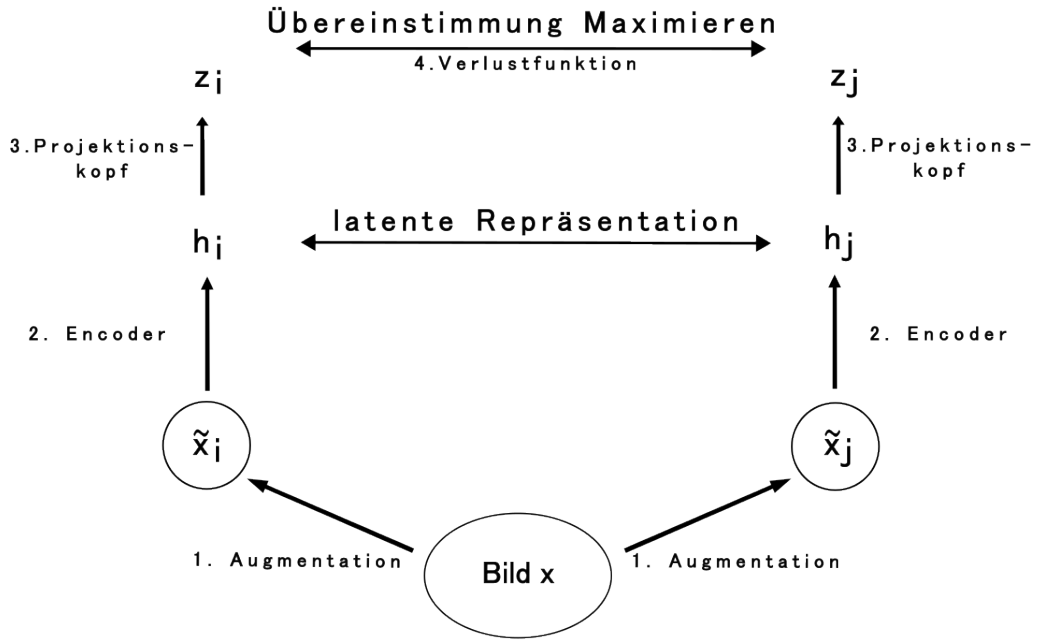


Abb. 2.2.: Schematische Darstellung eines kontrastiven Lernframeworks, mit einzelnen Schlüsselkomponenten (1-4). Aus dem Input x werden zwei Abbildungen durch die Augmentation erzeugt. Der Encoder projiziert diese dann in den latenten Raum, in dem dann durch den Projektionskopf und die Verlustfunktion die Übereinstimmung maximiert werden soll. Dies führt zur Entstehung eines initialen positiven Paars im Merkmalsraum. Abbildung in abgeänderter Form übernommen von Chen et al. [8].

3 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten zu Masked Autoencodern und Masked Image Modeling vorgestellt. Auf die detaillierte Struktur der einzelnen Modelle wird dabei bewusst verzichtet, da diese nicht zwingend für das Verständnis dieser Arbeit erforderlich ist. Grundlegende Kenntnisse über Transformer und Vision Transformer werden vorausgesetzt. Sollte dies nicht der Fall sein oder ein tieferes Verständnis angestrebt werden, empfehle ich die angegebenen Quellen sowie gegebenenfalls weiterführende Literatur.

3.1 BERT

Das grundlegende Prinzip des Masked Autoencoders stammt aus dem innovativen Paper und dem gleichnamigen Modell **“Bidirectional Encoder Representations from Transformers”** (BERT) [14], welches von Google 2019 vorgestellt wurde. BERT basiert dabei auf der Transformer-Architektur, verwendet jedoch nur den Encoder-Teil, (wie in Abbildung 3.1 erkennbar ist). Mithilfe von Maskierung wird eine starke bidirektionale Kontextverarbeitung möglich, das Modell lernt, den Satz von vorne und hinten zu verstehen und kann den Kontext des Satzes erkennen, was vorher nicht möglich war [@36, 14]. Die Trainingsmethode ist dabei sehr einfach: Ein zufälliges Token / Wort des Satzes wird maskiert und das Modell soll lernen, diesen Satz wieder zu rekonstruieren. Durch dieses Pretraining kann BERT anschließend für eine Vielzahl von Anwendungen feinjustiert werden, indem lediglich eine zusätzliche Ausgabeschicht hinzugefügt wird. Darunter zählt: Named Entity Recognition, Sentiment-Analyse, Fragebeantwortung und Textklassifikation [@36].

Bezug zu Masked Autoencodern: Die Funktionsweise von BERT entspricht einem Masked Language Modeling, welches eine Form des Masked Autoencoding ist, bei dem das Modell lernt, maskierte Teile der Eingabefrequenz zu rekonstruieren. Genau dieses Prinzip konnte später durch die Einführung der

Transformer in die Bildverarbeitung auch auf Bildern ähnlich angewendet werden. Daraus entstand der Masked Autoencoder. [17]

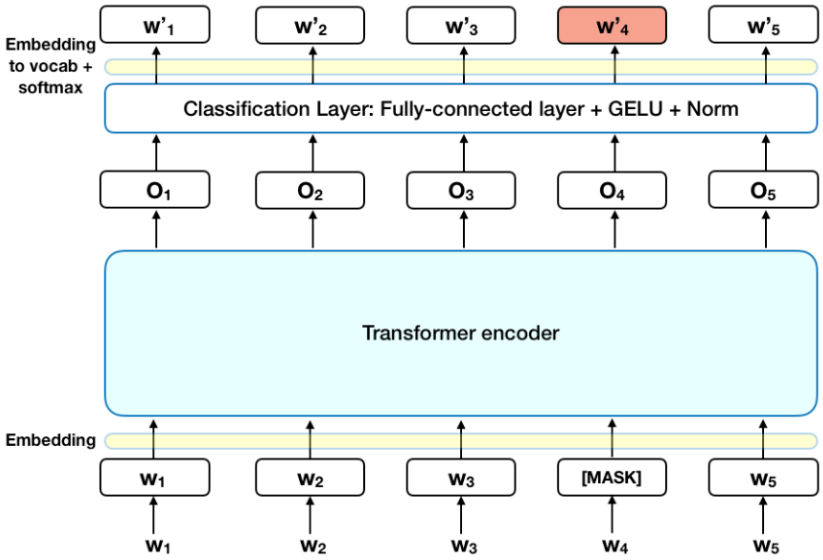


Abb. 3.1.: BERT Pre-Training Architektur. In dem Satz, welcher aus fünf Wörtern besteht, wird das vierte Wort maskiert, das Modell soll dann das maskierte Wort wieder richtig hervorsagen. Abbildung übernommen von Sumit Singh [@36]

3.2 Vision Transformer (ViT): Revolutionierung der Computer Vision

Die **Vision Transformer** (ViT), vorgestellt in dem bahnbrechenden Paper **"An Image is Worth 16x16 Words: Transformers for Image Recognitation at Scale"** von Dosovitskiy et al. (2020) [15], markieren einen Umbruch in der Computer Vision. Erstmals wurde die Transformer-Architektur, die ursprünglich für die natürliche Sprachverarbeitung entwickelt wurde, erfolgreich auf die Bildverarbeitungsaufgaben angewendet [15, @16].

Diese Arbeit stellt eine wichtige Weichenstellung dar, da sie demonstrierte, dass Convolutional Neural Networks (CNN) [28] nicht zwingend erforderlich für exzellente Leistungen in der Bildverarbeitung sind. Stattdessen kann ein Transformer, welcher direkt auf Sequenzen von Bildpatches angewendet wird, in der Bildklassifikation sehr gute Ergebnisse erzielen [15].

3.2.1 Architektur und technische Innovation

Die zentrale Innovation von ViT liegt in der Behandlung von Bildern als Sequenzen von Patches, ähnlich wie Transformer in der Sprachverarbeitung Wörter als Token behandeln [15, @16]. Ein Eingangsbild wird in gleich große, nicht überlappende Patches unterteilt, diese werden linearisiert und durch eine trainierbare lineare Projektion in einen höherdimensionalen Raum eingebettet (Embedding). Diese Patch-Embeddings werden mit gelernten Positionsembeddings kombiniert, um die räumliche Struktur des Originalbildes zu bewahren. [15]

Das Herzstück von ViT ist der Self-Attention-Mechanismus, der es ermöglicht, bereits in den untersten Schichten globale Beziehungen zwischen allen Bildpatches zu modellieren [15, @16]. Dies unterscheidet sich fundamental von CNNs, die lokale Merkmale durch hierarchische Schichten schrittweise zu globalen Repräsentationen aufbauen [@12].

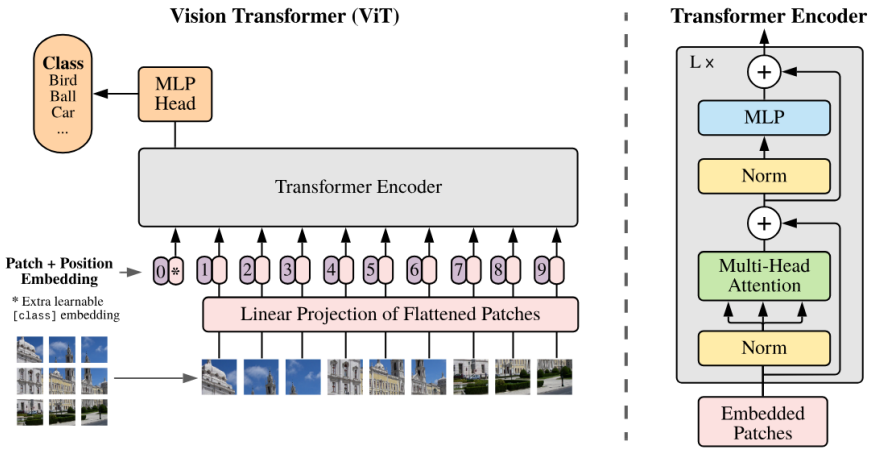


Abb. 3.2.: Architektur eines Vision Transformers; Abbildung übernommen von Dosovitskiy et al. [15]

3.3 Historie Masked Image Modeling

Masked Image Modeling hat sich in den vergangenen Jahren von einer Nischenidee zu einer weit verbreiteten Methode für das Self-Supervised-Pretraining entwickelt. Angefangen hat es womöglich mit Vincent et al. [39] im Jahr 2008,

die mit verzerrten Bildern versucht haben, einen Autoencoder zu trainieren [46], der sogenannte **Denoising Autoencoder** (DAE). Durch das Stapeln mehrerer DAE-Schichten konnten tiefe Netze robust initialisiert werden [39], die Bilder wurden also nicht wirklich maskiert, sondern nur etwas unkenntlich gemacht. Dieses Verfahren ebnete den Weg für weitere Forschungen.

Pathak et al. [32] erweiterten den Ansatz 2016 von einer pixelweisen Rekonstruktion hin zu einer semantischen Aufgabe und entwarfen den **Context Encoder**. Ein willkürliches Rechteck eines Bildes wird entfernt und das Netzwerk muss den Inhalt plausibel ergänzen [32]. Dabei stellte sich heraus, dass die erlernten Merkmale sehr gut als Vorinitialisierung für die Klassifikation funktionieren.

Der Durchbruch für die Maskierung kam dann in der Sprachverarbeitung mit BERT [14], dies zeigte, wie gut die Maskierung als Vortrainingsmethode funktioniert. Mit dieser bahnbrechenden Arbeit und der Einführung von Transformern für Bilder (ViT) [15] gab es einen regelrechten Schub für die Forschung in diesem Bereich.

Als erstes kam der **“Bidirectional Encoder representation from Image Transformers”** (BEiT) [2] als gut funktionierende Methode heraus, welche mithilfe eines discrete Variational Autoencoders (dVAE) Bilder in diskrete Tokens zerlegt und im Pretraining wieder rekonstruiert. Damit übertrifft BEiT erstmals überwachte ViT-Baselines auf ImageNet und demonstriert die Funktionsweise von BERT-ähnlichen Lernaufgaben im Bildraum [2].

Danach kamen die beiden Forschungsarbeiten über **Masked Autoencoder und SimMIM** heraus, welche in späteren Kapiteln nochmal genauer erklärt werden (Kapitel 3.4 und Kapitel 9). Der Masked Autoencoder unterscheidet sich von BEiT hauptsächlich in der Art der Rekonstruktion und der Architektur: Während beim Masked Autoencoder die maskierten Patches direkt auf Pixelebene rekonstruiert werden, nutzt BEiT einen extra trainierten Tokenizer und rekonstruiert somit die diskreten visuellen Tokens. Zusätzlich hat der Masked Autoencoder einen Decoder, was BEiT fehlt.

Self-distillated Masked Autoencoder (SdAE) [9] erweitert den Masked Autoencoder mit der Einführung eines Teacher-Student-Frameworks. Dabei rekonstruiert der Student nicht auf Pixelebene, sondern auf der latenten Feature-Repräsentation des Lehrers. Diese Art von Modellaufbau soll den Trainingsverlauf stabilisieren und die Lernfähigkeit steigern [9]. Eine genauere und

spezifischere Aufarbeitung kann in der Veröffentlichung von Chen et al. [9] nachgelesen werden.

Mit **Contrastive Masked Autoencoders** (CMAE) [19] wurde Ende 2023 nochmal ein großer Fortschritt im Bereich des Masked Image Modelings erzielt. CMAE vereinen die vorhandenen Stärken des Masked Autoencoders mit einem weiteren, kontrastiven Ziel. Das Modell versucht zunächst die Bilder zu rekonstruieren. Währenddessen werden aber zusätzlich Repräsentationen unterschiedlicher Ansichten desselben Bildes gegenübergestellt. Das heißt, die unterschiedlichen Ansichten der Bilder sollen im latenten Raum des kontrastiven Lernens nahe beieinander liegen. Dadurch lernt das Netzwerk, sowohl einzelne Bildinhalte präzise zu ergänzen, als auch robuste und gut unterscheidbare Merkmale der Bilder in seinem latenten Raum bereitzustellen. Diese Kombination ermöglicht es ihm noch eindeutigere und stärkere Bildrepräsentationen zu erzeugen, welche anschließend einen messbaren Vorteil in der Klassifikation bringen. [19, 18]

Eine umfassende Zusammenfassung der Historie ab 2021 ist in der Arbeit von Hondru et al. namens “Masked Image Modeling: A Survey” [18] zu finden. In dieser sind noch weitere Modelle, mit ausführlichen Erklärungen und Abbildungen, aufgelistet. Eine weitere Arbeit mit einer kurzen Zusammenfassung in der Related-Work-Sektion ist die Arbeit von Xu et al. [46].

3.4 Masked Autoencoder Are Scalable Vision Learners

Das Paper “Masked Autoencoder Are Scalable Vision Learners” von He et al. (2022) [17] knüpft an mehreren Forschungsarbeiten in der Computer Vision und dem Self-Supervised Learning an. Insbesondere an die bestehenden Arbeiten im Bereich des Masked Image Modeling und an Arbeiten aus dem Large Language Modeling-Bereich. [17]

Frühere Arbeiten im Bereich des Autoencoders (zum Beispiel Vincent et al., 2008 [39]) fokussierten sich hauptsächlich auf symmetrische Encoder-Decoder-Strukturen, bei denen Eingabedaten komprimiert und anschließend rekonstruiert werden. Das Masked Autoencoder-Modell erweitert dieses Konzept durch eine asymmetrische Architektur eines Autoencoders, in der kleine Teile des Bildes (Patches) maskiert werden. Dies hat den Rechenaufwand erheblich reduziert und führte zu einer besseren Downstream-Performance. [46]

Der Erfolg des Modells, so He et al., lässt sich auf die Maskierung zurückführen. Denn durch eine passende Maskierungsrate lernt das Modell semantisch komplexe Strukturen, statt auf Pixelebene zu lernen.

Der Masked Autoencoder wird später in Kapitel 4 weiter von mir behandelt werden und genauer erklärt, da meine Arbeit hauptsächlich auf dieser Modellstruktur basiert.

3.5 Weitere Forschungsarbeiten

Es gibt noch weitere Arbeiten, welche über die Verbesserung der Architektur von Masked Autoencodern handeln. Eines ist **“A Robustly and Effectively Optimized Pretraining Approach for Masked Autoencoder”** von Xu et al. (2023) [46]. In dieser Arbeit werden unterschiedliche Ansätze für die Verbesserung von Masked Autoencodern angesprochen, darunter zählt die dynamische Tokeninteraktion, welche in Kapitel 5.4.3 nochmal genauer untersucht wird, aber auch die Bedeutung von Pre-Patch-Normalisierung.

Es gibt auch Forschungen, die auf die Funktionsweise des Masked Autoencoders weiter eingehen, darunter zählt die Arbeit über den **“Uniformity-enhanced Masked Autoencoder”** (U-MAE) von Zhang et al. (2022) [49]. Diese stellt eine Verbindung zwischen MAE und kontrastivem Lernen her und zeigt, dass MAE implizit mask-induzierte positive Paare ausrichtet. Durch diese Erkenntnis konnte eine abgeänderte Architektur entwickelt werden, der sogenannte U-MAE, welcher das Problem des dimensionalen Kollapses in MAE adressiert.

4 Masked Autoencoder

In diesem Abschnitt wird ein Masked Autoencoder für das CIFAR-10-Datenset erstellt. Die Grundlage dafür bietet mit dem Article[1]: “Masked Autoencoders Are Scalable Vision Learners” [17] hauptsächlich das GitHub-Repository “MAE” von Xingyuan Zhang [@20]. Das hier beschriebene Modell wird im Verlauf der Arbeit untersucht und am Ende in Abschnitt 8.1, den Untersuchungsergebnissen entsprechend, angepasst.

4.1 Ansatz zum Masked Autoencoder

Ein Masked Autoencoder (wie er in Abbildung 4.1 zu sehen ist) funktioniert zunächst wie ein normaler Autoencoder. Er komprimiert mithilfe des Encoders eine Eingabe in die latente Repräsentation. Diese rekonstruiert er wieder mit dem Decoder. Der namensgebende Unterschied liegt hierbei in der Eingabe: Der Masked Autoencoder bekommt nicht das komplette Bild übergeben, sondern nur einen Teil des Bildes [17]. Wie dieses Teilbild aussieht, kann zwar selbst bestimmt werden, jedoch wird meistens die zufällige Maskierung mit einer Maskierungsrate von 75 % genommen [17, 45]. Umgesetzt wird das durch einen Vision Transformer (ViT) als Encoder, welcher das Bild mithilfe eines Convolutional Layers in einzelne, gleich große Patches unterteilt. Dabei wird aus jedem Patch ein Token erstellt und das Positions-Embedding darauf addiert [15]. Anschließend werden entsprechend der Maskierungsrate Patches maskiert, beziehungsweise gelöscht und durch die Transformer-Blöcke geschickt. Durch das Löschen der maskierten Patches werden diese nicht mehr weiter im Encoder behandelt. Dadurch ist das Netz nicht nur schneller, sondern erhält, wie die Arbeit von He et al. (2022) [17] zeigt, auch eine bessere Genauigkeit. [17]

Der Decoder versucht nun, das Bild aus der latenten Repräsentation möglichst genau zu rekonstruieren. Standardmäßig mit einer Loss-Funktion, welche auf Pixelwerten fungiert (z. B.: Mean Squared Error [@27]). Dabei ist der Input des

[1] Welcher hier auch immer gemeint ist, wenn ich Article schreibe

Decoders nun der vollständige Satz aus Bildern, bestehend aus den kodierten, sichtbaren Tokens und den maskierten Tokens. Die maskierten Tokens sind lernbare Vektoren mit einem Position-Embedding, damit der Decoder weiß, wo etwas fehlt. [17, 45, 48]

Das Ziel ist dasselbe wie bei BERT, nur mit Bildern. Der Encoder soll keine lokalen Muster erlernen oder Rückschlüsse auf benachbarte Pixel ziehen können, sondern das Bild semantisch so gut verstehen, dass es im späteren Verlauf für die Bilderkennung feinjustiert werden kann [17, 48]. Damit soll das Modell eine bessere Performance erreichen.

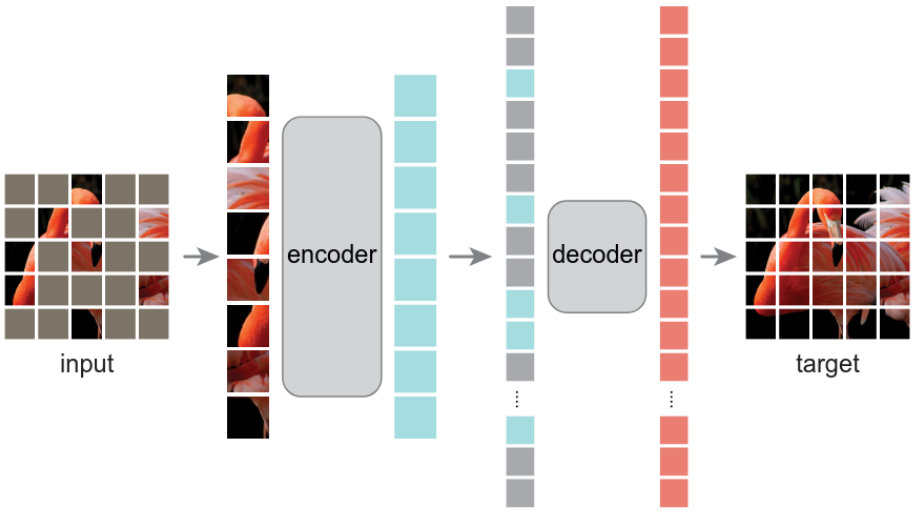


Abb. 4.1.: Architektur eines Masked Autoencoders; Abbildung übernommen von He et al. [17]

4.2 Architektur & Implementation von MAE

Nachdem jetzt ein Grundverständnis über Masked Autoencoder und deren Funktionsweise da sein sollte, wird genauer auf die Encoder-Decoder-Struktur eingegangen. Da ich keine große Rechenleistung besitze und eine passende Struktur für kleine Datensets erstellen will, musste ich einige Änderungen an dem eigentlichen Artikel vornehmen.

Encoder: Als Encoder habe ich eine ViT-Tiny [@40] ähnliche Struktur verwendet, da diese noch am ehesten für das CIFAR-10-Datenset verwendet werden

kann. Das heißt, mein Encoder hat eine Embedding-Dimension von 192 mit 12 Schichten und 3 Attention-Heads. Die Patch-Size wurde auf 2x2 Pixel angepasst, was insgesamt 16x16 (256) Patches bei 32x32 Pixel-großen Bildern entspricht. Die Maskierungsrate habe ich auf 75 % gesetzt und es werden nur die sichtbaren Patches in den Encoder übergeben. Mit dieser Konfiguration erhoffe ich gute Ergebnisse hinsichtlich Effizienz und Generalisierung.

Decoder: Der Decoder ist wesentlich kleiner als ein ViT-Tiny. Er hat nur 4 Schichten mit 3 Attention-Heads und einer Embedding-Dimension von standardmäßig 192. Jedoch wird er in Abschnitt 5.4 auch nochmal angepasst, um zu überprüfen, was die beste Decoder-Struktur für das Datenset ist. Was vielleicht auffällt, ist, dass die Struktur des Decoders kleiner ist als die des Encoders. Trotzdem ist die Architektur im Verhältnis (Encoder zu Decoder) zu der Implementierung im Artikel sehr groß [2]. Dies hat den Grund, dass der Decoder eine gewisse Tiefe benötigt, um die Spezialisierung auf die Rekonstruktionsaufgabe wirksam zu kapseln. Während die letzten Schichten eines Autoencoders typischerweise stark auf die Rekonstruktion von Pixeldetails spezialisiert sind, sind solche Details für nachfolgende Erkennungsaufgaben häufig weniger relevant [17]. Deshalb muss der Decoder eine ausreichende Tiefe haben, welche es ermöglicht, dass der Encoder abstraktere und allgemeinere Repräsentationen lernt, die sich besser für Klassifikations- oder Erkennungsaufgaben eignen [17].

Technische Implementierungsdetails: In der Implementation hat sich die Verwendung bestimmter Trainingstechniken für die Stabilisierung des Lernprozesses als entscheidend herausgestellt. Eine längere Warmup-Phase von 20 Epochen bei insgesamt 200 Epochen reduziert Instabilitäten, die durch die hohe Maskierungsrate entstehen. Das ermöglicht dem Optimizer, schrittweise eine stabile Lernrate zu finden und weniger auszuschweifen.

Der Einsatz von **AdamW mit Weight Decay (0.05)** nach Loshchilov & Hutter (2019) [29] trennt explizit Regularisierung und Parameterupdates, was insbesondere bei meiner begrenzten Rechenkapazität über Kaggle mit zwei T4 GPUs effektiv Overfitting entgegenwirkt. Durch die Verwendung von **Gradient Accumulation** wird trotz hardwarebedingter kleiner Batch-Größe eine hohe Batchsize von 4096 simuliert. Dies verbessert insbesondere die Konvergenzeigenschaften und die Gradientenschätzung [34].

[2] Im Verhältnis gesehen zum Encoder welcher im Article verwendet wurde

4.3 Trainingsverlauf

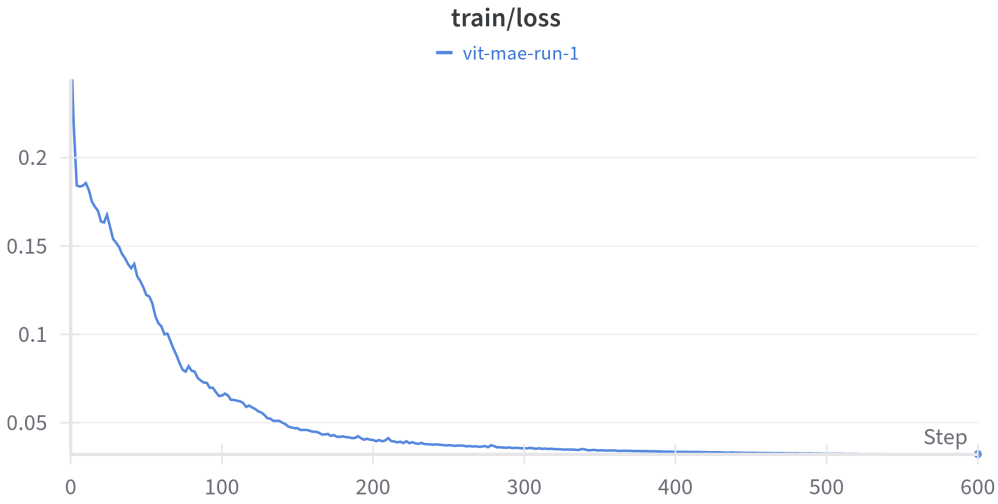


Abb. 4.2.: Trainingsverlauf über 300 Epochen / 600 Trainingsteps

Der dargestellte Trainingsverlauf 4.2 zeigt einen charakteristischen und vielversprechenden Verlauf. Man erkennt gut, dass das Modell schnell in der ersten Phase lernt. Der Loss fällt sehr schnell auf unter 0,05 bis zur 200. Epoche. Nach dieser Phase übernimmt die Feinabstimmungsphase, der Rückgang verlangsamt sich und nähert sich schlussendlich einem Loss von 0,03 an. Man kann also sagen, dass das Modell schon gut lernt und somit wahrscheinlich auch eine gute Rekonstruktion erzielt.

Die Rekonstruktionsbeispiele in Abbildung 4.3 bestätigen, dass das Modell sehr gut lernt, Bilder grob zu rekonstruieren. Lediglich die feinen Details in dem rekonstruierten Bild fehlen.

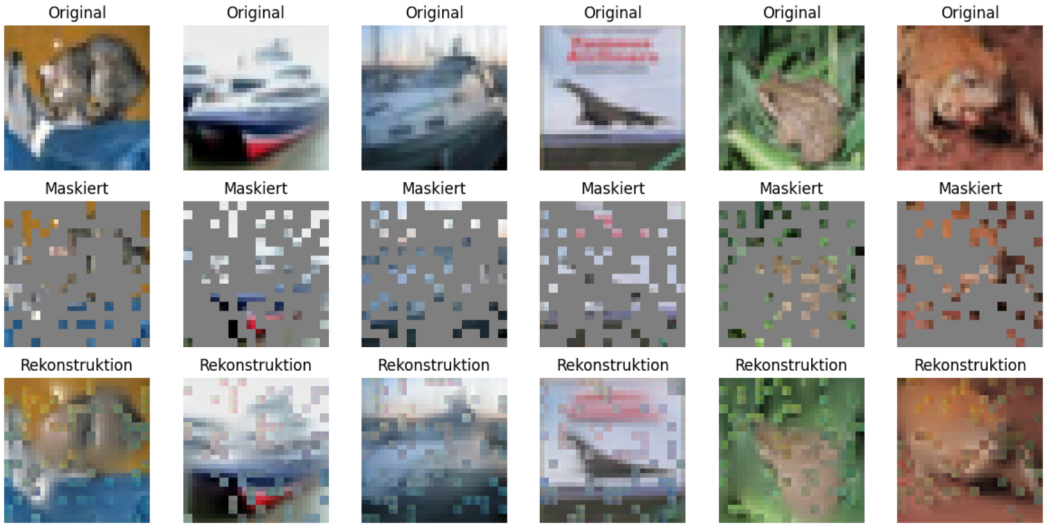


Abb. 4.3.: Rekonstruktion nach Training

4.4 Klassifikation

Nach dem Pretraining des Masked Autoencoders kann dieser für die Klassifikation fine-getunt werden. Das funktioniert, indem der Encoder aus dem Modell extrahiert wird und ein neuer Decoder erstellt wird, welcher für die Klassifikation vorgesehen ist und mit einem CLS-Token nach Delvin et al. (2019) [14] fungiert. Das Fine-Tuning arbeitet auf dem komplett sichtbaren Bild mit Labeln, also supervised. Als Optimizer kommt wieder AdamW zum Einsatz, kombiniert mit einem Cosine-Learning-Rate-Scheduler und einer Warmup-Phase. Die Anzahl der Lernepochen beträgt 30.

Um die Leistung besser beurteilen zu können, wird derselbe Encoder des MAEs nochmals implementiert, jedoch ohne weiteres Pretraining. Die zwei Modelle – eines ohne und eines mit Pretraining – werden anschließend unter denselben Trainingsbedingungen (weiter) trainiert. Ohne Pretraining wird eine Genauigkeit auf dem Test-Set von etwas über 79 % erreicht, mit dem Pretraining wird eine Genauigkeit von 84,5 % erreicht. Also eine Verbesserung von 5,5 %.

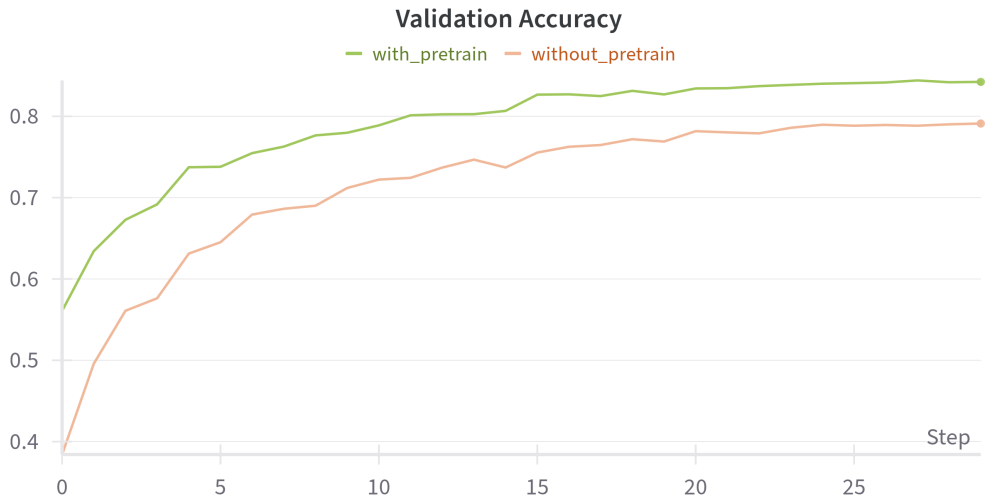


Abb. 4.4.: Genauigkeit auf dem Validation-Set, einmal mit und einmal ohne Pretraining. x-Achse: Anzahl der Epochen; y-Achse: Genauigkeit

Fazit Die Ergebnisse unterstreichen den Nutzen von Self-Supervised Prelearnings und zeigen, dass ein Masked Autoencoder in der Lage ist, hochinformative Merkmale für Downstream-Tasks zu extrahieren.

Die hier verwendete Architektur dient als Grundarchitektur und wird im nächsten Abschnitt 5 untersucht und verbessert.

5 Untersuchungen

Für die Untersuchungen werden wegen Performance-Gründen und Zeitknappheit für das Pretraining nur 100 Epochen und für das Fine-Tuning 30 Epochen verwendet, aber auch mit dieser stark limitierten Epochenanzahl werden noch gute Ergebnisse erzielt. Der Encoder wird nicht weiter untersucht, da ein größerer Encoder in der Regel bessere Ergebnisse erzielt, aber auch (wesentlich) mehr Trainingszeit benötigt [17]. Hier gilt es einfach, die Größe des Encoders an die Performance anzupassen.

5.1 Maskierungsrate

Das zentrale Element, welches einen Masked Autoencoder ausmacht, ist natürlich die Maskierung. Hier können viele unterschiedliche Versuche ausgeführt werden, unter anderem das Verhältnis zwischen unmaskierten und maskierten Bildpatches. Bei textbasierten Modellen wie BERT ist eine niedrige Maskierungsrate von circa 15 % üblich [14], jedoch trifft dies nicht zu für Masked Autoencoder und die anschließende Bildklassifizierung oder andere Downstream-Tasks, wie folgende Abbildung 5.1 zeigt.

Hierbei kann gut beobachtet werden, dass niedrige Maskierungsraten unter 50 % schlechter in der Klassifikation abschneiden als Maskierungsraten über die 60 % hinaus. Dennoch sind alle Ergebnisse besser als ohne Pretraining. Was jedoch unüblich ist, ist, dass die Maskierungsrate von 85 % so extrem besser abschneidet als eine Maskierungsrate von 75 %. Ein möglicher Grund für die besseren Ergebnisse dieser hohen Maskierungsrate liegt in der Funktionsweise von Masked Autoencoder. Bei höheren Maskierungsraten ist das Modell gezwungen, aus wenigen Daten und somit wenigen Informationen ein vollständiges Bild zu rekonstruieren. Dies führt zu einer besseren Erkennung globaler Bildstrukturen, da das Modell keine lokalen Details mehr trainiert oder aus den Nachbar-Patches des unmaskierten Patches direkt Verbindungen (weiter-)ziehen kann. [17]

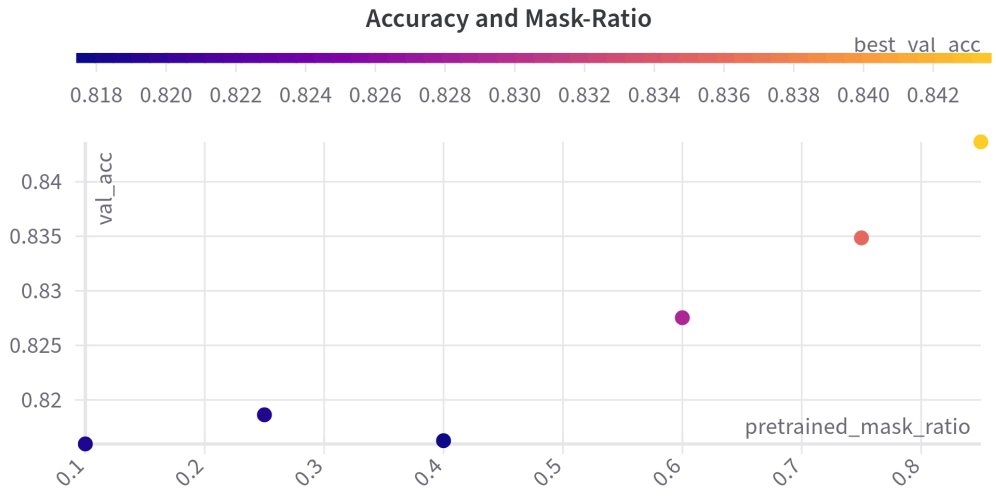


Abb. 5.1.: Rekonstruktion nach Training

Dieser Effekt kann vermutlich bei dem niedrig auflösenden CIFAR-10-Datensatz, mit klar abgegrenzten Klassen, nochmals verstärkt werden. Der Masked Autoencoder trainiert im Pretraining schon die wenigen Klassen sehr gut an, damit er eine halbwegs gute Rekonstruktion schafft. Dies geht umso besser mit wenigen, stark abweichenden Klassen wie in CIFAR-10.

Ob dieses ungewöhnliche Ergebnis jedoch konsistent ist oder doch nur ein Zufall, wird im nächsten Abschnitt 5.1.1 untersucht.

Als Anmerkung muss noch erwähnt werden, dass mit einer höheren Mask-Ratio der Autoencoder schneller trainiert, da die maskierten Patches nicht mit in den Encoder übergeben werden und dort nicht verarbeitet werden [17]. Dies ist besonders wichtig bei großskalierten Projekten.

5.1.1 Signifikanztest

Die Ergebnisse sind nach einem Durchlauf natürlich nicht signifikant oder repräsentativ, noch ist eine Aussage möglich, ob die Maskierungsraten überhaupt einen Einfluss haben. Deswegen habe ich das Experiment insgesamt viermal ausgeführt, um eine signifikante Aussage darüber treffen zu können. Einen Signifikanztest, wie ich ihn in diesem Abschnitt für die Maskierungsrate mache, werde ich nur hier machen. Es liegt leider nicht im Rahmen meiner Möglichkeiten, jede Untersuchung mehrmals laufen zu lassen.

Zuerst möchte ich eine Aussage darüber treffen, ob die Maskierungsrate überhaupt einen Einfluss hat. Den Test, den ich hierfür verwende, ist der ANOVA-Test; es handelt sich um eine Erweiterung des t-Tests auf mehr als zwei Gruppen.

Tab. 5.1.: Ergebnisse aller Durchläufe pro Maskierungsrate

Durchlauf	**25 %**	**60 %**	**75 %**	**85 %**
Durchlauf 1	0,8186	0,8275	0,8348	0,8437
Durchlauf 2	0,8188	0,8230	0,8237	0,8211
Durchlauf 3	0,8171	0,8254	0,8291	0,8311
Durchlauf 4	0,8179	0,8259	0,8300	0,8279

Die Maskierungsraten heißen im späteren Code nur Gruppe 1 bis Gruppe 4. Wichtig ist zu sehen, dass jede Gruppe eine Spalte in der Tabelle 5.1 darstellt.

ANOVA-Test

Voraussetzung: Für den ANOVA-Test müssen folgende Voraussetzungen [@38] gelten:

1. **Unabhängigkeit:** Die Messwerte sind alle vier unabhängig mit komplett neu trainierten Autoencodern trainiert worden und anschließend fine-getuned. Deshalb werden sie als unabhängig voneinander angenommen.

2. **Normalverteilung:** Innerhalb jeder Maskierungsrate sollten die Daten einer Normalverteilung folgen. Dies lässt sich mit dem Shapiro-Wilk-Test überprüfen. Zuerst werden Hypothesen aufgestellt:
   $H_0 =$ “Die Daten sind normalverteilt”;
   $H_A =$ “Die Daten sind nicht normalverteilt”.
   Bei einem Signifikanzniveau von 0,05 wurde der Test in R implementiert:

```r
# Shapiro-Wilk-Test fuer jede Gruppe durchfuehren
ergebnisse <- list(
  "Gruppe 1" = shapiro.test(gruppe1),
  "Gruppe 2" = shapiro.test(gruppe2),
  "Gruppe 3" = shapiro.test(gruppe3),
  "Gruppe 4" = shapiro.test(gruppe4)
)

print(ergebnisse)
```

P. 5.1: Shapiro-Wilk-Test

Tab. 5.2.: Ergebnisse des Shapiro-Wilk-Test für jede Gruppe

Gruppe	**0,25 %**
Gruppe 1 (25 %)	0,5789
Gruppe 2 (60 %)	0,8346
Gruppe 3 (75 %)	0,8678
Gruppe 4 (85 %)	0,7769

Die Ergebnisse in Tabelle 5.1 zeigen, dass alle Werte über dem Signifikanzniveau von 0,05 liegen und somit die Nullhypothese der Normalverteilung nicht verworfen wird.

3. **Varianzhomogenität:** Für jede Maskierungsrate müssen die Varianzen innerhalb der Gruppe ungefähr gleich sein [@38]. Dies lässt sich gut mit dem Levene-Test überprüfen. Zuerst wird wieder eine Null-Hypothese H_0 und eine Alternativ-Hyptohese H_A aufgestellt [@5]:
$H_0 =$ *"Die Varianz zwischen den Maskierungsraten ist ähnlich";*
$H_A =$ *"Die Varianz zwischen den Maskierungsraten ist nicht ähnlich"*.
Als Signifikanz-Wert habe ich wieder 0,05 gewählt und das Ganze in R mithilfe der Bibliothek "car" implementiert:

```r
# Datenrahmen erstellen
daten <- data.frame(
    werte = c(gruppe1, gruppe2, gruppe3, gruppe4),
    gruppe = factor(rep(1:4, each = 4))
)
# Levene-Test durchfuehren
library(car)
leveneTest(werte ~ gruppe, data = daten)
```

P. 5.2: Levene-Test

Das Ergebnis ist ein p-Wert von $p = 0,127$, was größer $0,05$ ist. Damit ist die Varianzhomogenität gegeben und der ANOVA-Test kann ausgeführt werden.

Durchführung ANOVA: Die Ausführung des eigentlichen Tests mache ich etwas ausführlicher als die vorherigen beiden Tests. Als Erstes wird wieder die Nullhypothese und die Alternativhypothese aufgestellt:
$H_0 =$ "Alle Gruppenmittelwerte sind gleich."
$H_A =$ "Mindestens ein Gruppenmittelwert weicht signifikant von den anderen ab."
Untersucht habe ich $k = 4$ Gruppen mit jeweils $n_i = 4$ Messwerten. Damit

komme ich auf insgesamt $N = 16$ Messwerte. Des Weiteren ist x_{ij} der j-te Messwert in Gruppe i, $\bar{x}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_{ij}$ das Gruppenmittel der Gruppe i und $\bar{x} = \frac{1}{N}\sum_{i=1}^{k}\sum_{j=1}^{n_1} x_{ij}$ das Gesamtmittel.

Als erstes wird das Gruppen- und Gesamtmittel berechnet:

$$\begin{aligned}
\bar{x}_1 &= \frac{0,8186 + 0,8188 + 0,8171 + 0,8179}{4} = 0,81810,\\
\bar{x}_2 &= \frac{0,8275 + 0,8230 + 0,8254 + 0,8259}{4} = 0,82545,\\
\bar{x}_3 &= \frac{0,8348 + 0,8237 + 0,8291 + 0,8300}{4} = 0,82940,\\
\bar{x}_4 &= \frac{0,8437 + 0,8211 + 0,8311 + 0,8279}{4} = 0,83095,\\
\bar{x} &= \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4} x_{ij} = 0,825975.
\end{aligned}$$

Als nächstes berechnen wir die Quadratsummen zwischen Gruppen (SSB) und innerhalb der Gruppen (SSW):

$$\begin{aligned}
\mathrm{SSB} &= \sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2\\
&= 4\Big[(0,81810 - 0{,}825975)^2 + (0,82545 - 0{,}825975)^2\\
&\quad + (0,82940 - 0{,}825975)^2 + (0,83095 - 0{,}825975)^2\Big]\\
&= 0,000395
\end{aligned}$$

$$\begin{aligned}
\mathrm{SSW} &= \sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2\\
&= [\ldots]\\
&= 0,000343
\end{aligned}$$

Die Freiheitsgrade sind folgendermaßen:

$$df_{\text{between}} = k - 1 = 3, \quad df_{\text{within}} = N - k = 12.$$

Mithilfe dieser können die mittleren Quadratsummen berechnet werden.

$$\begin{aligned}
\mathrm{MSB} &= \frac{\mathrm{SSB}}{df_{\text{between}}} = \frac{0,000395}{3} = 0,00013167,\\
\mathrm{MSW} &= \frac{\mathrm{SSW}}{df_{\text{within}}} = \frac{0,000343}{12} = 2,8583 \times 10^{-5}
\end{aligned}$$

Schließlich kann mit den ganzen Werten die F-Statistik und damit der zugehörige p-Wert berechnet werden:

$$F = \frac{\text{MSB}}{\text{MSW}} = \frac{0,00013167}{2,8583 \times 10^{-5}} = 4,60658, \qquad p = 1 - F_{\text{CDF}}(F; 3, 12) \approx 0,0229.$$

Da $p = 0,0229 < 0,05$ ist, wird die Nullhypothese verworfen. Damit weicht mindestens ein Gruppenmittelwert statistisch signifikant ab, und somit hat die Maskierungsrate einen Einfluss.

Konfidenzintervall

Wie in Tabelle 5.1 abzulesen ist, hat eine Maskierungsrate von 85 % das beste Ergebnis erreicht. Aber zugleich hat sie auch teilweise schlechter als die 75-prozentige Maskierungsrate abgeschnitten. Aus diesem Grund möchte ich nochmal diese zwei Maskierungsraten genauer untersuchen und berechne dafür die durchschnittliche Genauigkeit, die Standardabweichung und deren Konfidenzintervalle (KI). Auch wenn ein Konfidenzintervall mit lediglich vier Stichproben nicht besonders aussagekräftig ist, ist mehr leider nicht möglich. Als Konfidenzniveau habe ich eine 95 % t-Verteilung wegen der geringen Anzahl an Durchführungen gewählt.

Tab. 5.3.: Durchschnitt und Standardabweichung der Genauigkeit

Maskierung	Mittelwert	Standardabw.	KI
75 %	0,82940	0,0046	[0,8221, 0,8367]
85 %	0,83095	0,0095	[0,8158, 0,8461]

Durchschnittliche Genauigkeit & Standardabweichung: Zu erkennen ist, dass eine Maskierungsrate von 85 % bessere Ergebnisse erreichen kann, aber auch eine höhere Standardabweichung hat und somit auch schlechter als die 75-prozentige Maskierungsrate abschneiden kann. Wenn konsistente Ergebnisse gewünscht sind, sollte man nicht höher als diese Maskierungsraten gehen. Eine Untersuchung mit einer Maskierungsrate von 95 % hat keine Verbesserung mehr gegeben, sondern die Genauigkeit ist stark eingebrochen.

5.2 Maskierungsart

Eine weitere Untersuchung im Bereich der Maskierung ist die Art und Weise der Maskierungstechnik. Als nützlichste (siehe Abbildung 5.2) Maskierungsart hat sich die zufällige Maskierung herausgestellt, es gibt jedoch auch noch andere Maskierungsarten, wie:

1. Blockweise Maskierung
2. Muster-Maskierung (z.B. Gittermuster, Schachbrettmuster, etc.)
3. Progressive Maskierung (also eine steigende Maskierungsrate pro Epoche)
4. Semantisch geleitete Maskierung

Untersuchen werde ich in erster Linie nur die Gittermaskierung, die blockweise Maskierung, die zufällige Maskierung und die progressive Maskierung. Dies hat den Grund, dass bei allen Muster-Maskierungen eine ähnliche Performance zu erwarten ist [44] und die semantisch geleitete Maskierung schwer für mich umsetzbar ist. Für die semantisch geleitete Maskierung müsste ein eigenes Netz trainiert werden, welches die Objekte der Bilder lokalisieren kann, was jedoch nicht die einzige Schwierigkeit darstellt. Das Objekt würde mit den großen, maskierten Patches direkt komplett maskiert sein und es kann keine Klassifizierung ohne Objekt stattfinden. Die Patches kann ich leider auch nicht kleiner machen, wie in dem späteren Abschnitt 5.3 gezeigt wird.

5.2.1 Ergebnisse & Interpretation

Die Rekonstruktionen mit der jeweiligen Maskierungstechnik sind in dem Appendix A.1 angehängt. Was die Klassifikation angeht, hat die zufällige Maskierung klar am besten abgeschnitten. Die Muster- und blockweise Maskierung hat bei einer Maskierungsrate von 60 % eine maximale Genauigkeit von ca. 82 % erreicht und die progressive Maskierung hat mit knapp unter 82 % tatsächlich am schlechtesten abgeschnitten, obwohl der Trainingsverlauf sehr vielversprechend aussah.

Die Auswertung der unterschiedlichen Strategien zeigt deutlich, dass die zufällige Maskierung hinsichtlich der Klassifikation von Bildern der anderen

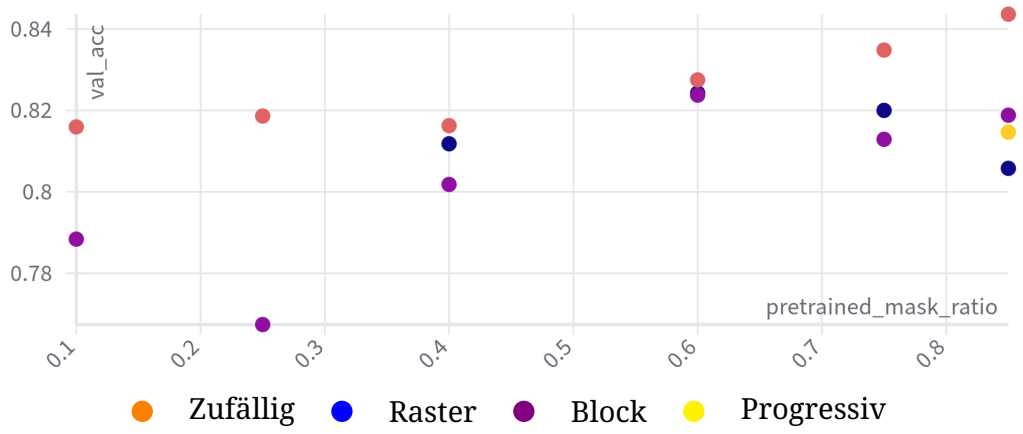


Abb. 5.2.: Accuracy mit jeweiliger Mask-Ratio und Strategie

Tab. 5.4.: Best erreichte Genauigkeit für jede Strategie

Strategie	Maskierungsrate	Genauigkeit
Zufällig	85 %	84,37 %
Block	60 %	82,38 %
Muster	60 %	82,43 %
Progressiv	0 - 85 %	81,47 %

untersuchten Strategien überlegen ist. Dies deckt sich auch mit den Erkenntnissen der Originalarbeit von He et al. (2022) [17]. Das lässt sich vermutlich darauf zurückführen, dass bei der zufälligen Maskierung die Verteilung der maskierten Bereiche über das gesamte Bild hinweg variabel ist. Dadurch ist das Modell eher gezwungen, kontextuelle Informationen aus einer Vielzahl unterschiedlicher Regionen zu nutzen und schlussendlich gut zu generalisieren. Denn nur so kann der Decoder eine vernünftige Rekonstruktion liefern.

Im Gegensatz dazu sind bei der blockweisen oder muster-basierten Maskierung die maskierten Bereiche oft zusammenhängend oder regelmäßig verteilt. Dies könnte dazu führen, dass das Modell eher lokale Muster oder spezifische Strukturen lernt zu rekonstruieren, statt ein Verständnis für das ganze Bild zu erlangen, was später für die Klassifikation wichtig wäre. Gerade bei der blockweisen Maskierung und insbesondere bei gering auflösenden Bildern können auch die Hauptmerkmale des Bildes komplett fehlen, was zu einer sehr erschwerten Rekonstruktionsaufgabe führt, welche schlecht Kontextinformationen erlernen kann. Was auch wieder mit verwandten Arbeiten in diesem Gebiet übereinstimmt, wie die Studien von Xie et al. (2022) [44], die auch zeigen, dass alternative Maskierungsstrategien wie blockweises Maskieren keine Vorteile bringen.

Die progressive Maskierung, bei der die Maskierungsrate stetig gesteigert wird, zeigte zunächst im Trainingsverlauf vielversprechende Ergebnisse, mit einem geringen Validation Loss und einer hohen Startgenauigkeit. Dennoch blieb die maximale Genauigkeit hinter jeder anderen Strategie zurück. Ein möglicher Grund könnte sein, dass das Modell zu Beginn des Trainings mit zu vielen sichtbaren Informationen arbeitet und deswegen eher auf einfache Rekonstruktionsaufgaben spezialisiert ist, auch wenn sie später schwerer werden. Jedoch hat eine progressive Maskierungsrate zwischen 50 und 85 Prozent auch nur 81 % Genauigkeit erreicht. Deshalb könnte der Grund hierfür noch weiter untersucht werden.

5.2.2 Fazit

Zusammenfassend lässt sich festhalten, dass die Wahl der Maskierungsstrategie einen erheblichen Einfluss auf die Lernfähigkeit und die finale Klassifikationsgenauigkeit des Modells hat. Die zufällige Maskierung stellt dabei die effektivste Methode dar und ist gleichzeitig auch noch mit die schnellste im Pretraining. Dies liegt an ihrer hohen Diversität an Lernaufgaben, welche es nicht möglich macht, lokale Muster zu lernen und somit globale Muster gelernt werden müssen. Die anderen untersuchten Strategien können zwar in bestimmten Szenarien sinnvoll sein, erreichen im Allgemeinen jedoch eine geringere Performance.

Für zukünftige Arbeiten könnte es noch interessant sein, hybride Ansätze zu untersuchen, bei denen verschiedene Maskierungsstrategien kombiniert werden, oder auch die semantisch geleitete Maskierung, in der das Hauptthema des Bildes mehr maskiert wird.

5.3 Patch-Größe

Die Patch-Größe beschreibt die Größe der nicht maskierten Flächen. Mit einer kleinen Patch-Größe gibt es viele kleine sichtbare Punkte, mit einer großen Patch-Größe sind es wenige, dafür große sichtbare Felder auf dem Bild. Ich werde vier unterschiedliche Patch-Größen untersuchen: 1, 2, 4 und 8.

5.3.1 Ergebnisse & Interpretation

Meine Untersuchungen haben überraschende Ergebnisse gezeigt. Eine etwas größere Patch-Größe von 4 hat mit weitem Abstand am besten abgeschnitten. Mit einer Genauigkeit von über 85 % konnte ich die Performance um einiges verbessern, gerade auch, weil eine größere Patch-Größe schneller trainiert. Der Pretraining-Durchlauf mit einer Patch-Größe von 1 hat fast viermal so lange gebraucht wie der Durchlauf mit einer 2-er Patchgröße, weswegen ich keine weiteren Experimente mit einer kleinen Patch-Größe durchführen kann.

Tab. 5.5.: Genauigkeit je Patch-Größe

Patch-Größe	**Genauigkeit**
1	75,94 %
2	83,54 %
4	85,23 %
8	76,19 %

Die Ergebnisse der Untersuchung stimmen auch mit dem Größenverhältnis der Patch-Größe der Originalarbeit [17] überein. Fundierte Gründe für diese Erkenntnis kann ich jedoch nicht geben. Es ist möglich, dass der MAE zu wenig aus kleinen Patches extrahieren kann und größere Sichtfelder benötigt. Andererseits lernt er bei großen Patches eher aus benachbarten Pixeln Rückschlüsse zu ziehen, statt die grundlegende Struktur des Bildes zu erkennen.

5.4 Decoder

Der Decoder ist ein wichtiger Bestandteil des Masked Autoencoders, auch wenn er oft weniger Beachtung als der Encoder erfährt. Während der Encoder die sichtbaren Patches verarbeitet und zu der latenten Repräsentation umformt, verarbeitet der Decoder die latente Repräsentation und versucht, möglichst genau das ursprüngliche Bild zu rekonstruieren.

Da es jedoch schlussendlich nicht auf die Qualität der Rekonstruktion ankommt, sondern darum, dass der Encoder ein grundlegendes Verständnis der Bilder erlangt, spielt der Decoder nur eine unterstützende Rolle.

In dieser Arbeit werden zwei unterschiedliche Decoder-Hyperparameter untersucht. Einmal die Decoder-Tiefe (also die Anzahl an Layers), sowie die Embedding-Dimension des Decoders. Darüber hinaus wird in Abschnitt 5.4.3 noch ein Dropout auf die Attention-Matrix angewendet, um Oversmoothing zu vermeiden.

5.4.1 Decoder Tiefe

Die Tiefe des Decoders, gemessen an der Anzahl der Transformer-Layer, hat einen größeren Einfluss auf die Leistung des Pretrainings. Die Ergebnisse der Untersuchung zeigen, dass für den CIFAR-10-Datensatz eine Decoder-Tiefe von 4 optimal ist.

Bei einer geringen Tiefe von nur einem Layer liegt die Validierungsgenauigkeit bei circa 82,77 %. Mit zunehmender Tiefe verbessert sich aber die Leistung um etwa 1 % auf 83,54 %, bei einer Tiefe von 4 Layern. Werden nun noch mehr Layer hinzugefügt, fällt die Genauigkeit wieder drastisch; bei 8 Layern liegt sie nur noch bei knapp unter 81 %.

Diese Ergebnisse lassen sich folgendermaßen interpretieren (Vermutung):

1. Ein zu flacher Decoder hat möglicherweise nicht genügend Kapazität, um die komplexen Rekonstruktionen durchzuführen.

2. Ein zu tiefer Decoder neigt eher zu Overfitting und verschlechtert damit die Generalisierungsfähigkeit.

3. Ein Decoder von etwa 4 Layern bietet ausreichende Expressivität für die Rekonstruktionsaufgabe und der damit verbundenen Klassifikation.

Diese Beobachtungen sind auch maßgeblich entscheidend für den Erfolg des Masked Autoencoders, denn wenn nur tiefe Decoder funktionieren würden, würde das Training wesentlich länger dauern. Darüber hinaus zeigt das Ergebnis auch, dass die Genauigkeit der Klassifikation nicht an möglichst guten Rekonstruktionen gebunden ist.

5.4.2 Embedding-Dimension

Die Embedding-Dimension des Decoders ist ein weiterer wichtiger Hyperparameter, welcher die Leistung des Masked Autoencoders beeinflusst. Untersucht wird die Embedding-Dimension des Decoders, denn diese muss in einer asymmetrischen Encoder-Decoder-Struktur nicht der Embedding-Dimension des Encoders entsprechen. Dabei werden vier unterschiedliche Dimensionen untersucht: $48, 96, 192, 384$.

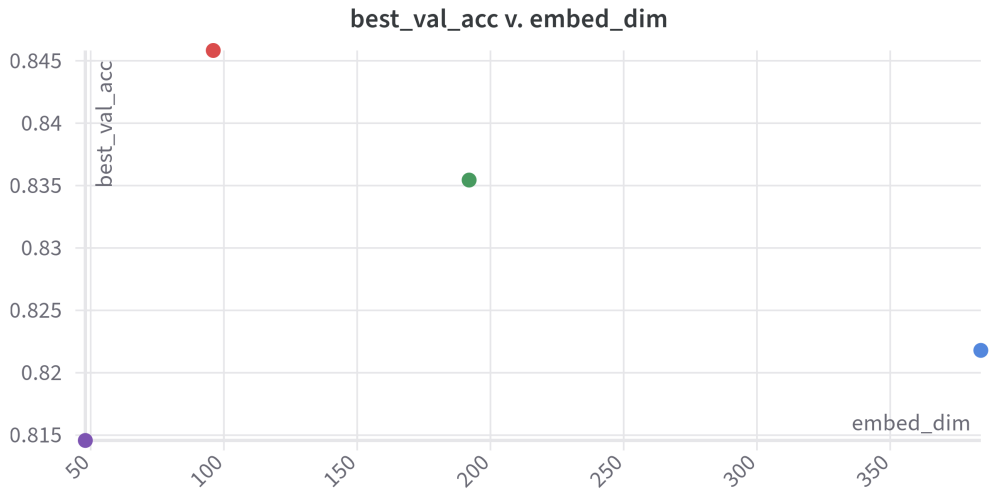


Abb. 5.3.: Genauigkeit bei entsprechender Embedding-Dimension

Die Beobachtungen, welche aus Abbildung 5.3 hervorgehen, dass eine Embedding-Dimension von circa 96 optimal ist, könnten sich wieder durch folgende Faktoren erklären lassen:

1. Eine zu niedrige Embedding-Dimension von 48 bietet eine nicht ausreichende Repräsentationskraft um die komplexen semantischen Informationen der Encoder-Ausgabe angemessen zu verarbeiten.

2. Eine zu hohe Embedding-Dimension von über 200 führt zu einer Überparametrisierung des Decoders, was das Risiko von Overfitting erhöht.

3. Die optimale Dimension liegt bei circa 96; dies stellt das optimale Gleichgewicht zwischen Ausdrucksstärke und Effizienz dar.

Wie auch bei der Decoder-Tiefe ist eine kleine Embedding-Dimension für die Effizienz des Systems nützlich.

Diese Erkenntnisse unterstreichen erneut, dass ein vergleichsweise kleiner Decoder effektives Lernen ermöglicht.

5.4.3 Dynamische Token-Interaktion

Das Paper “A Robustly and Effectively Optimized Pretraining Approach for Masked Autoencoder” von Xu et al. (2023) [46] beschäftigt sich mit der Optimierung des Pretrainings von Masked Autoencodern. Eine Verbesserung ist, dass im Decoder auf die Attention-Matrix Dropout verwendet wird. Dadurch werden zufällige Token (hier Aufmerksamkeitsgewichte) maskiert, was die Entstehung kontextueller Repräsentationen fördern soll und Oversmoothing vermeidet [46], was eine dynamische Token-Interaktion darstellt.

Der Hintergrund dieser Methode entsteht daraus, dass eine diverse Token-Interaktion gefördert werden soll. Ohne Dropout neigen Attention-Mechanismen dazu, sich auf wenige dominante Token zu konzentrieren, was zu einer übermäßigen Spezialisierung führt [50]. Durch das zufällige Maskieren von Attention-Gewichten wird das Modell dazu gedrängt, Informationen aus verschiedenen kontextuellen Bereichen zu extrahieren, anstatt sich auf eine begrenzte Anzahl von Token zu verlassen [46]. Es dient als eine stochastische Regularisierungsmethode, welche Overfitting und Oversmoothing vermeiden soll [1].

Umsetzung: Die Umsetzung dieser Technik erfolgt durch das gezielte Anwenden von Dropout direkt auf die Matrix der Self-Attentions im Decoder. Konkret bedeutet das, dass nach der Berechnung der Attention-Scores mittels der Gleichung [46]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.1}$$

ein Dropout-Layer auf die normalisierte Softmax-Ausgabe angewendet wird. Das führt dazu, dass einzelne Einträge mit einer bestimmten Wahrscheinlichkeit einfach auf Null gesetzt werden. Diese künstlich erzeugte Störung soll den Autoencoder dazu bewegen, Informationen aus variierenden Teilausschnitten des kontextuellen Raums zu extrahieren, anstatt sich auf alle Token gleichzeitig verlassen zu können. [46]

Durch diese Technik wird das Modell gewissermaßen mit einer dynamisch reduzierten Sicht auf die Eingabestruktur konfrontiert, was einem partiell verbundenen Graphen im Sinne von Graph Neural Networks (GNNs) entspricht [46]. Dies erhöht die Robustheit der gelernten Repräsentationen und unterbindet zugleich, dass der Decoder durch übermäßige Konvergenz auf mittlere Werte (Oversmoothing) an Ausdruckskraft verliert [46].

Implementiert ist das dann folgendermaßen:

```
class AttentionWithDropout(nn.Module):
    def init(self, dim, num_heads=NUM_HEADS, attn_drop=ATTN_DROP):
    super().init()
    self.scale = (dim // num_heads) ** -0.5
    self.qkv = nn.Linear(dim, dim * 3)
    self.attn_drop = nn.Dropout(attn_drop)  # Dropout auf Scores
    self.proj = nn.Linear(dim, dim)
    def forward(self, x):
        B, N, C = x.shape
        q, k, v = self.qkv(x).chunk(3, dim=-1)
        attn = (q @ k.transpose(-2, -1)) * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)  # Maskierung

        x = (attn @ v).transpose(1, 2).reshape(B, N, C)
        return self.proj(x)
```

P. 5.3: Attention-Layer mit Dropout im Decoder

Ergebnisse & Interpretation

Tab. 5.6.: Accuracy mit Dropout und ohne Dropout im Decoder

Decoder	Genauigkeit
ohne Dropout	83,48 %
mit Dropout	84,16 %

Neben der Verbesserung der Genauigkeit von über 0,68 % trainiert das Netz auch etwas schneller. Diese zwei Beobachtungen sprechen klar für die Verwendung von dynamischen Token-Interaktionen und somit der Verwendung von Dropout-Layern auf der Attention-Matrix. Das liegt an der verbesserten Generalisierungsfähigkeit des Modells.

5.4.4 Fazit

Die Untersuchung der Decoder-Architektur im Masked Autoencoder offenbart klare Designprinzipien, welche für das effiziente Lernen nötig sind.

Ein Decoder mit vier Transformer-Layern (die Tiefe) und 96-dimensionalen Embeddings erwies sich als optimaler Kompromiss zwischen Modellkapazität und Recheneffizienz und bietet dabei noch die beste Leistung. Darüber hinaus verhindert eine dynamische Tokeninteraktion Overfitting und führt zu einer besseren Performance.

5.5 Datenaugmentation

Die Datenaugmentation ist normalerweise eine zentrale Technik im Deep Learning, um die Generalisierungsfähigkeit von Modellen zu verbessern und Overfitting zu reduzieren [47]. Mithilfe von Data Augmentation gelingt es, die Trainingsdatensätze mittels Bildtransformationen[1] künstlich zu erweitern. Dadurch hat das Modell mehr Trainingsdaten und ist zusätzlich robuster gegenüber Variationen in den Eingabedaten [47]. In meiner Arbeit habe ich verschiedene Augmentierungsstrategien untersucht und deren Einfluss auf die Modellperformance beleuchtet.

Für die Experimente wurden folgende Augmentierungen implementiert:

- **Random Crop:** Mit `transforms.RandomResizedCrop((32, 32), scale=(0.5, 1.0))` werden zufällige Ausschnitte aus dem Bild extrahiert und auf die Zielgröße 32x32 Pixel skaliert. Diese Methode simuliert verschiedene Bildausschnitte und soll dem Modell helfen, relevante Merkmale, unabhängig von der Position im Bild, zu erkennen. Es kann jedoch auch vorkommen, dass das Objekt nicht mehr im Bild ist und somit eine Klassifikation stark erschwert wird.
- **Crop + ColorJitter:** Mit `transforms.RandomResizedCrop((32, 32))` wird zuerst wieder Random Crop angewendet. Zusätzlich wird nun noch die Helligkeit, Kontrast und Sättigung zufällig verändert (`transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2)`). Diese Kombination sorgt nicht nur für räumliche, sondern auch farbliche Variabilität in den Lerndaten.

[1] Wie Colorjittern oder Resizen

- **Spiegelung:** Spiegelt/flippt das Bild horizontal oder vertikal.
- **Keine Augmentierung:** Als Vergleichsbasis dient ein Modell komplett ohne Augmentation, also wird auf den rohen Trainingsdaten trainiert.

5.5.1 Ergebnisse & Interpretation

Die Evaluation der Trainingsdurchläufe zeigt auf den ersten Blick ungewöhnliche Unterschiede in der Klassifikationsleistung.

Tab. 5.7.: Ergebnisse der Untersuchung zur Data Augmentation

Augmentierung	**Genauigkeit**
Random Crop	80,42 %
Crop + ColorJitter	81,64 %
Spiegeln	82,87 %
Keine Augmentierung	83,54 %

Wie zu erkennen ist, ist keine der untersuchten Augmentations von Vorteil. Sie schneiden sogar schlechter ab als keine Augmentierung. Das steht im starken Kontrast zu anderen Deep Learning Verfahren, in welchen Augmentierung kaum wegzudenken ist [47]. Über die Gründe, warum ColorJitter und Random Crop schlechter als keine Augmentierung abschließen, lässt sich nur mutmaßen:

1. **Informationsverlust:** Das Cropping kann möglicherweise kontextuelle Schlüsselmerkmale entfernen. Damit geht der Kontext des Bildes verloren und das Modell kann diesen somit nicht mehr lernen.
2. **Modellkapazität:** Flache Netzwerkarchitekturen profitieren weniger von starken Augmentierungen [45].
3. **Einfache Textmuster:** Gerade ColorJitter könnte für einfache Bilder mit geringer Auflösung, wie in CIFAR-10, kontraproduktiv sein.

5.5.2 Fazit

Daraus lässt sich schließen, dass gerade für kleine Datensätze mit geringer Auflösung keine Augmentation verwendet werden sollte und wenn doch, nur sehr leichte Augmentierung, wie ein horizontaler Flip oder Ähnliches.

5.6 Loss Funktion

Bei Masked Autoencodern soll der Loss nur auf den maskierten Patches fungieren, da dies das Modell zwingt, semantische Kontextinformationen aus den sichtbaren Bereichen abzuleiten, statt triviale Kopien zu lernen und zugleich noch rechenressourcenschonender optimiert werden kann. Deshalb wird auf diese Weise auch eine leicht höhere Genauigkeit erzielt. [17]

Dies war in den bisherigen Untersuchungen von mir folgendermaßen implementiert:

```
1 loss = torch.mean((predicted_img - img) ** 2 * mask) / mask_ratio
```

Dabei wird zunächst der klassische Mean Squared Error (MSE) berechnet, welcher anschließend mit der Maske multipliziert wird. Die Maske enthält die Werte 0 (sichtbar) und 1 (maskiert), sodass nur der Fehler in den maskierten Regionen berücksichtigt wird. Abschließend wird dieser Wert durch die Maskierungsrate `mask_ratio` dividiert und der Mittelwert dieser Berechnung genommen. Dies hat den Zweck, den durch die hohe Maskierungsrate sehr kleinen Loss in einen realistischen und normalisierten Loss-Wert zurückzuwandeln.

Es gibt aber auch noch andere, vielleicht bessere, Loss-Funktionen. Ich habe drei Loss-Funktionen herausgesucht, welche aus verschiedensten Gründen vielversprechend sind:

Mean Absolute Error: Der Mean Absolute Error ist ähnlich zu dem Mean Squared Error (MSE), jedoch ist er nicht quadratisch, sondern linear im Fehler und somit weniger ausreißerempfindlich [@6]. Mathematisch lässt sich der Mean Absolute Error wie folgt ausdrücken:

$$\mathrm{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{5.2}$$

Dabei bezeichnet y_i den tatsächlichen Wert, $\hat{y}_i$ die Vorhersage und n die Anzahl der Datenpunkte [@6].

Huber-Loss: Der Huber-Loss stellt dagegen einen Kompromiss zwischen dem Mean Squared Error und dem Mean Absolute Error dar [@7]. Besonders vorteilhaft ist er, wenn das Modell empfindlich auf Ausreißer im Datensatz

reagiert. Während MSE Ausreißer stark gewichtet, da der Fehler quadriert wird, gewichtet der Huber-Loss bei großen Abweichungen nur linear, also ähnlich zum Mean Absolute Error und bei kleinen Fehlern ähnlich zu MSE [@7, @35]. Die Loss-Funktion [@35] sieht dann folgendermaßen aus:

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{if } |y - f(x)| \leq \delta \\ \delta\left(|y - f(x)| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{5.3}$$

Seine Eigenschaften machen ihn potenziell auch gegenüber stark abweichenden Pixelwerten in schwer rekonstruierbaren Bereichen robust. Einzelne Bereiche im Bild, welche aufgrund starker Maskierung kaum rekonstruierbar sind, dominieren so den Trainingsverlauf weniger.

Structural Similarity Index Measure: Als drittes und letztes habe ich den Structural Similarity Index Measure (SSIM) untersucht. Es ist eine perceptuelle Metrik, welche die strukturelle Ähnlichkeit zwischen Original und Rekonstruktion auf Basis von Helligkeit, Kontrast und Textur bewertet [51]. Mathematisch wird SSIM durch folgende Gleichung [51] beschrieben:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{5.4}$$

wobei μ_x und μ_y die Mittelwerte, σ_x und σ_y die Standardabweichungen der Bilder x und y sind, und σ_{xy} deren Kovarianz bezeichnet. Die Konstanten c_1 und c_2 dienen der Stabilisierung der Division bei kleinen Nennerwerten.

SSIM ist aus dem Grund interessant, weil es nicht mehr die pixelgenauen Differenzen minimiert, sondern versucht, eine strukturelle Konsistenz herzustellen. Es ist also ein anderer Ansatz, welcher sehr oft für Bildvergleiche herangezogen wird, wie allein schon die Anzahl an Zitaten des Ursprungspaper [51] in anderen Papern zeigt (über 40.000).

5.6.1 Ergebnisse & Interpretation

Meine Ergebnisse zeigen, dass der Mean Squared Error der effektivste Loss (zumindest für das CIFAR-10-Datenset) zu sein scheint.

Tab. 5.8.: Ergebnisse der Untersuchung zur Data Augmentation

Loss-Funktion	Genauigkeit
MSE	83,54 %
MAE	82,68 %
Huber-Loss	81,75 %
SSIM	81,49 %

Mit einer Genauigkeit von 82,68 % liegt der Mean Absolute Error etwas hinter dem MSE. Die etwas komplizierteren Loss-Funktionen haben erstaunlicherweise am schlechtesten abgeschnitten.

Ein Grund dafür könnte sein, dass der MSE empfindlicher auf größere Fehler reagiert und somit stärker zum Erlernen feiner Unterschiede im Bildinhalt beiträgt. Gleichzeitig deuten die relativ geringen Unterschiede zwischen den Loss-Funktionen darauf hin, dass das Modell insgesamt robust gegenüber der Wahl der Verlustfunktion ist.

Interessanterweise konnte der auf die Wahrnehmungsähnlichkeit ausgerichtete SSIM-Funktion nicht mit den klassischen fehlerbasierten Loss-Funktionen mithalten. Dies lässt vermuten, dass SSIM eher für Aufgaben geeignet ist, bei denen visuelle Ähnlichkeiten aus menschlicher Perspektive wichtig sind und nicht aus der Sicht eines Computers.

5.6.2 Fazit

Insgesamt zeigt sich, dass die Wahl der Loss-Funktion zwar einen Einfluss hat, dieser jedoch keinen drastischen Einfluss auf die Klassifikationsleistung darstellt. Da der MSE sehr leicht zu implementieren ist und die beste Genauigkeit in meiner Untersuchung liefert, ist er die beste Wahl.

6 Kritische Hinterfragung meiner Untersuchung

Die gewonnenen Erkenntnisse meiner Untersuchungen zu der Maskierungsrate / -art, Decoder-Architekturen und Data-Augmentation-Strategien liefern zwar wertvolle Einblicke für den CIFAR-10-Datensatz, unterliegen jedoch methodischen und datenspezifischen Limitierungen, gerade im Kontext des Overfittings.

Overfitting & Limitierung von CIFAR-10: Vision Transformer benötigen extrem viele Daten, um diese nicht zu overfitten und gut zu generalisieren [48]. Dabei sind selbst Datensätze wie ImageNet [13] mit mehreren Hunderttausend gelabelten Bildern teilweise zu klein beziehungsweise nicht gut genug gelabelt [3]. CIFAR-10 mit 50.000 Trainingsdaten kann dementsprechend auch für kleine Vision-Transformer-Architekturen wie den ViT-Tiny zu klein sein. Er overfittet die Daten sehr schnell und hat somit eine schlechte Performance auf neuen Daten [52]. Dagegen musste ich stark in meinen Untersuchungen ankämpfen, was bei größeren Datensätzen nicht so extrem der Fall gewesen wäre.

Des Weiteren bringt CIFAR-10 noch mehr Einschränkungen mit sich, welche sich auf den Trainingsverlauf mit einem Masked Autoencoder auswirken. Zum einen die niedrig auflösenden Patches mit nur 4x4 Pixeln, die gegebenenfalls nur wenig semantisch sinnvolle Strukturen erfassen, was die Fähigkeit des Transformers beeinträchtigt. Dies müsste weiter untersucht werden. Zum anderen die klare Klassenabgrenzung mit lediglich zehn Objektklassen, da das Modell hier vielleicht nur einfache Schlüsselmerkmale wie den Farbverlauf oder die Hintergrundfarbe lernen muss, um eine Vorhersage zu treffen, statt kontextuelle Muster zu lernen.

Komplexität: Die Komplexität eines Masked Autoencoders ist sehr hoch. Meist benötigen komplexere Systeme mehr Daten und mehr Trainingszeit. Mein Fokus liegt jedoch auf einem kleinen Datenset, weswegen der Masked

Autoencoder vielleicht nicht die beste Wahl ist. Dennoch gelingt ein Einblick in das Thema und es werden grundlegende Designstrukturen klar, welche für den Erfolg maßgebend sind.

Signifikanz: Meine Untersuchungen wurden, bis auf wenige Ausnahmen, alle nur einmal durchgeführt. Dies liefert keine zuverlässigen Ergebnisse. Darauf sollte beim Lesen der Untersuchungen geachtet werden.

7 Vertiefung der Funktionsweise

Der Masked Autoencoder lernt, wie in den vorherigen Abschnitten zu sehen ist, inhärente Strukturen des Bildes. In diesem Kapitel werde ich genauer auf die Gründe und die Rolle der Maskierung eingehen. Mit diesem Thema haben sich auch schon mehrere Paper beschäftigt, wie das von Zhang et al. (2022) [49] und von Kong et al. (2023) [25].

Die grundlegende Hypothese hinter MAE beruht auf der Annahme, dass durch das (zufällige) Maskieren großer Bildteile das Modell gezwungen wird, semantische Repräsentationen zu erlernen, anstatt sich auf die oberflächlichen Textureigenschaften oder lokalen Interpolationen zu verlassen [17]. Diese Annahme hat sich durch die Tests bestätigt, benötigt jedoch eine tiefere theoretische Erklärung, um die Mechanismen des Masked Autoencoders vollständig zu ergründen.

7.1 Rolle der Maskierungsrate

Die Maskierungsrate ist ein wichtiger Parameter bei dem Trainieren eines Masked Autoencoders und geht weit über einen Regularisierungsmechanismus hinaus. Der Einfluss kann besser und mathematisch untersucht werden, indem die zugrunde liegende Datenstruktur als hierarchisches latentes Variablenmodell dargestellt wird [25].

Das geht als Graph: Sei $G := (V, E)$ ein Graph, wobei die Knoten V aus den Pixeln (X) und den latenten Variablen (Z) bestehen. Sprich: $V := (X, Z)$. Während des Trainings des Netzwerks können beliebige latente Variablen des Graphen gelernt werden. Der Graph ist als Baum in Abbildung 7.1 dargestellt.

Die Blätter des Baumes sind die Pixel des Bildes (X). Die inneren Knoten und die Wurzeln zeigen die latenten Variablen (Z). Dabei sind die Variablen, welche näher am Knoten sind, High-Level-Repräsentationen, wie zum Beispiel

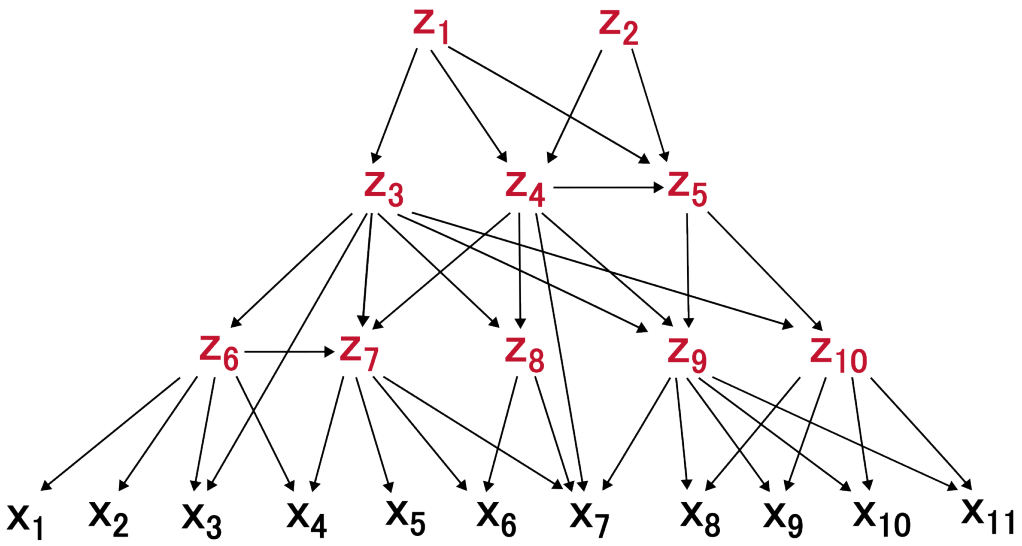


Abb. 7.1.: Hierarchischer Graph eines Bildes mit Pixeln $x_1 - x_{11}$ und den lernbaren latenten Variablen $z_1 - z_{10}$. Dabei sind die oberen latenten Variablen die High-Level-Repräsentationen. Bild von Kong et al. [25] nachempfunden

die Klasse des Bildes. Weit entfernte Variablen, beziehungsweise Variablen nah an den Blättern, sind Low-Level-Repräsentationen, wie Texturen. Das Ziel, welches wir mit einem Masked Autoencoder verfolgen, ist es, die High-Level-Repräsentationen zu erlernen. Dabei hat das Maskierungsverhältnis einen Einfluss, denn: ein Modell mit geringer Maskierungsrate kann nur low-level latente Repräsentationen (wie Texturen) erlernen, dasselbe gilt für sehr hohe Maskierungsraten. Man sucht also den Sweet-Spot dazwischen. [25, 17]

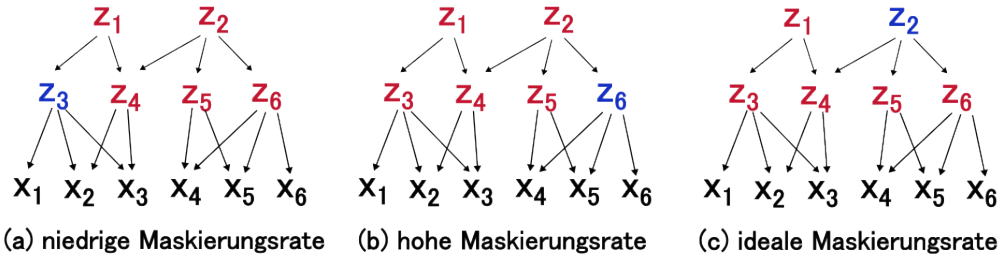


Abb. 7.2.: Unterschiedliche Maskierungsraten (aufsteigend von links) und deren gelernte latente Repräsentation welche blau markiert ist. Bild von Kong et al. [25] nachempfunden

Dabei haben die Autoren Kong et al. mit zwei Theoremen mathematisch beweisen können, dass MAEs unter bestimmten Annahmen tatsächlich latente Variablen im hierarchischen Modell identifizieren können. Sie zeigen,

dass für jede spezifische Maske eine entsprechende minimale Menge latenter Variablen existiert, die die statistische Abhängigkeit zwischen dem maskierten und dem sichtbaren Teil vollständig erfasst. Diese theoretische Garantie erklärt, warum MAEs hochwertige semantische Repräsentationen aus niedrigstufigen Pixelmerkmalen extrahieren können. Die Theoreme werde ich hier nicht aufzeigen. [25]

Dies spiegelt sich auch in meinen Untersuchungen über die Maskierungsrate in Kapitel 5.1 wider. Dort sieht man genau, wie moderate Maskierungsraten zwischen 50-80 % die beste Performance erreichen und somit sehr wahrscheinlich eine hohe hierarchische latente Repräsentation gelernt haben.

7.1.1 Untersuchung / Experiment

Kong et al. [25] haben diese Theorie aber auch nochmal experimentell überprüft. Für die Überprüfung haben sie für unterschiedliche Mask-Ratios den SSIM und den MSE zwischen Originalbild und rekonstruiertem Bild berechnet. Dabei ist das Ziel zu zeigen, dass bei niedriger und sehr hoher Maskierungsrate der MSE sehr gering ist, da das Modell niedrige Repräsentationen lernt, welche auf Pixel-Ebene fungieren. Bei einer passgenauen Maskierungsrate (wie 75 %) soll dafür der MSE relativ hoch sein. Für den SSIM ist es genau umgekehrt. Er soll bei ausgefallenen Maskierungsraten sehr gering sein (geringe Ähnlichkeit, hoher Loss) und bei der passgenauen Maskierungsrate sehr hoch sein (also starke Ähnlichkeit, geringer Loss). Da der SSIM strukturelle Ähnlichkeit zwischen Original und Rekonstruktion auf Basis von Textur bewertet, soll das Experiment zeigen, dass der Masked Autoencoder bei etwas höheren Maskierungsraten gute semantische Strukturen erfassen kann. Nochmal zusammengefasst:

- Niedriger MSE & niedriger SSIM → Low-Level-Repräsentation erlernt
- Niedriger MSE & hoher SSIM → Ungenaues Ergebnis, eher High-Level-Repräsentation (sollte im Normalfall nicht vorkommen)
- Hoher MSE & niedriger SSIM → Generell schlechtes Ergebnis
- Hoher MSE & hoher SSIM → High-Level-Repräsentation erlernt

Diese Annahme kommt daher, dass SSIM die Ähnlichkeit eines Bildes anhand von Strukturen berechnet. Ist diese Ähnlichkeit sehr hoch, so muss das Modell wissen, was auf dem Bild abgebildet ist und hat somit ein tiefes semantisches

Wissen. Anders sieht dies bei dem MSE Loss aus, dieser fungiert auf Pixelebene und muss deshalb nicht allzu viel Wissen über das Bild haben und somit weniger semantisches Wissen. [25]

Ich habe dieses Experiment auch für den CIFAR-10-Datensatz nachgebaut und ausgeführt (siehe Bild 7.3). Meine Ergebnisse entsprechen trotz zweifacher Ausführung nicht den Erwartungen entsprechend und widersprechen den Ergebnissen der Originalarbeit [25]. Die SSIM-Ähnlichkeit ist wie erwartet bei einer Maskierungsrate von circa 75 % am höchsten und bei außergewöhnlichen Maskierungsraten geringer. Dies spricht für die Behauptung, dass höhergradige latente Repräsentationen bei passender Maskierungsrate gelernt werden. Die Untersuchung über den MSE-Loss widerspricht jedoch dieser Annahme, denn dieser ist bei niedrigen Maskierungsraten sehr hoch und bei den angesprochenen 75 % am geringsten.

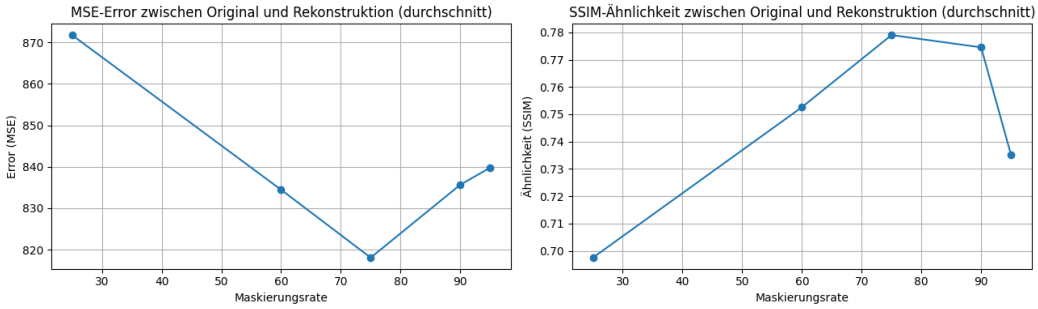


Abb. 7.3.: Rekonstruktionsvalidierung; Ergebnisse des Experiments, welche Repräsentationen gelernt werden.

Den Grund für dieses unterschiedliche Ergebnis kann ich nur mutmaßen. Ich gehe davon aus, dass die Pixelanzahl von 32x32 Pixeln in einem CIFAR-10-Bild einfach zu gering ist, um einen gegenläufigen Verlauf zwischen SSIM und MSE zu erzeugen. Das Modell lernt zu schnell gute semantische Strukturen, was den MSE bei der geringen Pixelanzahl direkt mit minimiert. Aber dies sind alles nur Mutmaßungen und haben keine wissenschaftliche Grundlage!

7.2 Verbindung zu kontrastivem Lernen

Eine weitere bemerkenswerte theoretische Erkenntnis über Masked Autoencodern liegt in der formalen Verbindung zwischen MAE und kontrastivem

Lernen. Die Verbindung mag zuerst nicht offensichtlich sein, da der Masked Autoencoder auf Rekonstruktion basiert, während kontrastives Lernen explizit positive und negative Beispielpaare verwendet und daraus einen Merkmalsraum bildet. Dennoch lässt sie sich mathematisch aufzeigen und gibt neue Einblicke in die Funktionsweise. Denn: durch die Maskierung entstehen implizite positive Paare im Sinne von kontrastivem Lernen [49].

7.2.1 Motivation

Bevor es mit den mathematischen Details anfängt, eine kurze Erklärung, warum diese Verbindung überhaupt relevant ist. Kontrastives Lernen gilt als eine erfolgreiche Methode im Self-Supervised-Learning-Umfeld, da es semantisch bedeutsame Repräsentationen lernt. Wenn eine Verbindung zwischen Masked Autoencoder besteht, erklärt dies, warum das Maskieren so erfolgreich ist. Darüber hinaus ermöglicht es auch, theoretische Garantien aus dem kontrastiven Lernen auf MAE zu übertragen.

Die zentrale Hypothese dabei ist: Ein MAE erzeugt durch das Maskieren implizierte positive Paare, die dann durch die Rekonstruktionsaufgabe ausgerichtet werden. Also genau wie kontrastives Lernen auch die expliziten positiven Paare ausrichtet.

7.2.2 Intuitive Erklärung

Für die intuitive Erklärung der Verbindung nehme ich zwei unterschiedliche Bilder eines Objektes, hier ein Auto. Wenn beide Bilder stark maskiert sind, bleiben möglicherweise nur noch ähnliche Bereiche übrig, wie zum Beispiel Reifen, Scheinwerfer oder Karosserieteile. Das MAE-Modell lernt, diese ähnlichen Regionen auf ähnliche Weise zu repräsentieren [49].

Warum und wie entstehen positive Paare, nach dem kontrastiven Lernen, durch Maskierung aber genau?
Die Maskierung mit einem hohen Verhältnis (circa 75 %) bewirkt zwei wichtige Dinge:

1. **Reduktion von störenden Details:** Durch das Ausblenden vieler Patches, bleiben vor allem relevante Merkmale sichtbar, was zu weniger Varibilität innerhalb einer Klasse führt. [49]

2. **Förderung der intra-klassen Ähnlichkeit:** Objekte derselben Klasse weisen nach der Maskierung mit hoher Wahrscheinlichkeit ähnliche sichtbare Regionen auf, dies erhöht die Ähnlichkeit der Bilder. Was die Konnektivität im zugrundeliegden Graphen erhöht, welcher gleich genauer erklärt wird. [49]

7.2.3 Mathematische Verbindung

Die theoretische und mathematische Grundlage dieser Verbindung basiert auf der Konstruktion zweier Graphstrukturen:

1. Der **Mask Graph** G_M, ein bipartiter Graph zwischen unmaskierten Patches $x_1 \in X_1$ und allen maskierten Patches aus $x_2 \in X_2$. Wobei die Kantengewichte durch die gemeinsame Wahrscheinlichkeit $M(x_1, x_2) = \mathbb{E}_{\bar{x}}[M(x_1, x_2|\bar{x})]$ definiert sind, also die Wahrscheinlichkeit, dass x_1 und x_2 aus einem gemeinsamen Bild stammen und durch die Maskierung aufgeteilt wurden ($\bar{x}$ ist das zugrundeliegende Bild; $M(x_1, x_2|\bar{x})]$ die Indikatorfunktion, ob x_1 und x_2 komplementäre Views desselben Inputs sind). [1] [49]

   Der Graph zeigt den Lernprozess des MAEs, indem er aufzeigt, wie maskierte Patches genutzt werden, um die unmaskierten Patches zu rekonstruieren. Er erfasst die direkte Input-Output-Beziehung des MAE-Lernprozesses. [49]

2. Der **Augmentation Graph** G_A. Dieser modelliert die Verbindungen zwischen allen unmaskierten Ansichten in X_1, welche durch das Masking entstehen. Dabei werden die Kantengewichte zwischen zwei Ansichten $x_1, x_1^+ \in X_1$ folgendermaßen definiert: $\mathbb{E}_{x_2} M(x_1|x_2)M(x_1'|x_2)$. Also die Wahrscheinlichkeit, dass zwei verschiedene unmaskierten Ansichten dasselbe Rekonstruktionsziel x_2 haben. Wenn die Wahrscheinlichkeit hoch ist, wären die beiden Ansichten x_1 und x_1^+ in dem Modell ein positives Paar. [49]

   Der Graph erfasst also die impliziten positiven Paare. Ansichten, die durch verschiedene Masken desselben Bildes entstanden sind. Dies stellt den Schlüssen in der Verbindung zu kontrastivem Lernen dar.

[1] Stark zusammengefasste Version, für eine ausführlichere Ausführung siehe [49].

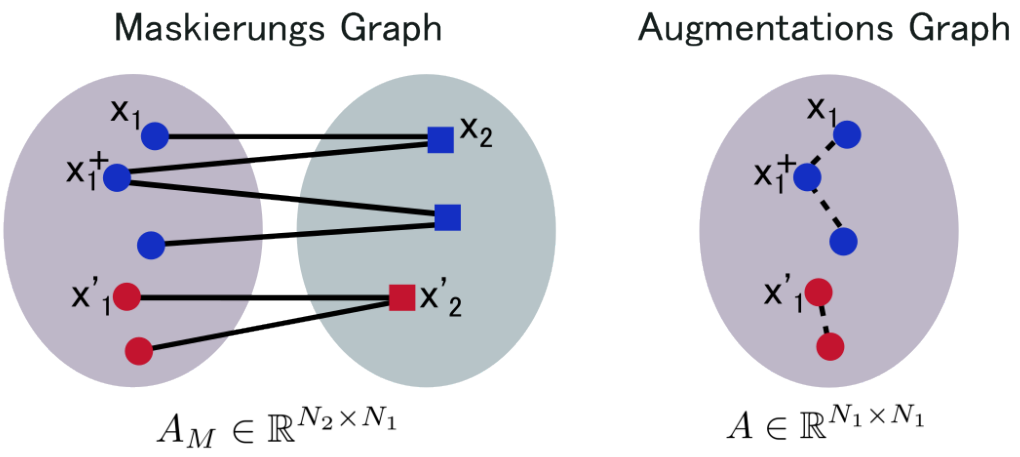


Abb. 7.4.: Die Abbildung zeigt den bipartiten Maskierungs Graph und den Augmentations Graphen. Leicht abgeändert von [49] übernommen

Von der Rekonstruktion zu der Ausrichtung

Mithilfe dieser Graphen leitet Zhang et al. zwei zentrale Theoreme ab, die den MAE-Verlust nach unten durch einen Ausrichtungsverlust begrenzen:

1. Das erste zentrale Theorem beschränkt den MAE-Verlust durch einen Ausrichtungsverlust auf dem Maskierungs Graph nach unten.

$$L_{MAE}(h) \geq L_{asym}(h) - \epsilon + \text{const} \tag{7.1}$$

   wobei $L_{asym}(h) = -\mathbb{E}_{x_1,x_2}\left[h(x_1)^T h_g(x_2)\right] = -\operatorname{tr}\left(H_g^T \bar{A_M} H\right)$ [49] (Die Ähnlichkeit zwischen Encoder-Ausgabe von x_1 und Decoder-Ausgabe x_2). Das heißt, ein kleiner MAE-Verlust impliziert eine gute Ausrichtung zwischen unmaskierten Eingaben und deren rekonstruierten maskierten Pendants [49]. Anders gesagt, ist der MAE-Verlust klein, so ist das Original und die rekonstruierte Version gut ausgerichtet. Der MAE lernt die Features von verbundenen Knoten im Maskierungsgraph zusammenzubringen.

2. Das zweite Theorem verbindet die beiden definierten Graphen. Zhang et al. zeigen, dass der asymmetrische Ausrichtungsverlust (aus Theorem 1) auf dem Maskierungs Graph durch den symmetrischen Ausrichtungsverlust auf den Augmentation Graph nach unten begrenzt werden kann [49]:

$$L_{\text{asym}}(h) \geq \frac{1}{2} L_{\text{align}}(h) + \text{const} \tag{7.2}$$

mit $L_{\text{align}}(h) = -\mathbb{E}_{x_1, x_1^+}\left[h(x_1)^T h(x_1^+)\right]$, wobei (x_1, x_1^+) positive Paare entsprechend dem Augmentations-Graphen sind. Dies lässt sich folgendermaßen verstehen: Wenn zwei unmaskierte Ansichten (x_1, x_1^+) das Selbe maskierte Ziel haben (x_2), so bewirkt das Ausrichten von x_1 an x_2 und von x_1^+ an x_2, dass auch x_1 und x_1^+ näher kommen. Daher entsteht im Augmentationsgraph ein implizietes positives Paar, das ausgerichtet wird. Dieses Theorem stellt die Grundlage der Verbindung zu kontrastives Lernen auf. Den formalen Beweis lasse ich hier weg, dieser kann aber in der Originalarbeit [49] nachgeschaut werden.

Durch die Kombination beider Theoreme erhalten wir folgende Aussage [49]:

$$L_{\text{MAE}}(h) \geq \frac{1}{2} L_{\text{align}}(h) - \varepsilon + \text{const} \tag{7.3}$$

Diese Aussage zeigt (ohne Beweis), dass ein MAE und kontrastives Lernen das Gleiche tun. Sie richten (positive) Paare aus. Ein kleiner MAE-Rekonstruktionsfehler impliziert, dass die Merkmale positiver Paare stark ausgerichtet sind. Insgesamt zeigen die Theoreme, dass der MAE-Trainingsprozess einem kontrastiven Ziel folgt. [49]

Der Unterschied zwischen MAE und kontrastivem Lernen ist, dass kontrastives Lernen positive Paare durch explizite Datenaugmentation erzeugt und MAE diese durch Maskieren und Rekonstruieren erstellen kann.

7.2.4 Grund für die hohe Maskierungsrate

Warum geht dies aber nur mit einer hohen Maskierungsrate und nicht mit einem normalen Autoencoder?

Einerseits führt eine höhere Maskierung zu mehr Verbindungen im Augmentation Graph, da ähnliche unmaskierte Regionen wahrscheinlicher werden, es gibt also eine **höhere Konnektivität**. Aber auch die **semantische Gruppierung** wird stärker, denn Objekte derselben Klasse werden durch die Reduktion auf wesentliche Merkmale stärker zusammengebracht. Es ist also die **Balance zwischen Signal und Rauschen** auf welche es bei dem Maskieren ankommt. Zu wenig Maskierung führt zu wenigen positiven Paaren, zu viel Maskierung zerstört semantische Informationen.

7.2.5 Zusammenfassung

Als kleine Zusammenfassung nochmal die Unterschiede und Gemeinsamkeiten von Masked Autoencodern und kontrastivem Lernen [49]:

Gemeinsamkeiten:

- Die Ausrichtung positiver Paare.
- Ähnliche mathematische Struktur im Ausrichtungsverlust.
- Die theoretischen Garantien aus dem konstrastivem Lernen lassen sich auch auf Masked Autoencoder anwenden.

Unterschiede:

- Die Entstehung positiver Paare. Der Masked Autoencoder verwendet Maskierung, kontrastives Lernen verwendet explizite positive Paare durch Datenaugmentation.
- Die Architektur. Der Masked Autoencoder hat einen Decoder, der als flexible Projektionskomponente fungiert, kontrastives Lernen hingegen hat meist nur einen einfachen Encoder mit Projektionskopf.

8 Fazit Masked Autoencoder

8.1 Schlussendliches Ergebnis

Schlussendlich bleibt die Frage offen, was für eine Performance der Masked Autoencoder mit allen Verbesserungen auf dem CIFAR-10-Datensatz hat. Folgende Parameter habe ich für meinen letzten Durchlauf verwendet:

Tab. 8.1.: Parameter des optimierten Durchlaufs

Parameter	**Wert**
Maskierungsrate	80 %
Maskierungsart	zufällige Maskierung
Patch-Größe	4
Decoder-Tiefe	4
Embedding-Dimension	96
Dynamische Token-Interaktion	Ja
Data-Augmentation	keine
Loss Funktion	MSE
Epochen	300
Warmup-Epochen	60

Mit einer Top-Genauigkeit von 90,53 % ist meine verbesserte Pretraining-Struktur um 6 % besser als die ursprüngliche Struktur in Kapitel 4.4. Ein Ergebnis von circa 90 % ist auf CIFAR-10 schon sehr gut und kann mit einem Vanilla-ViT-Tiny (max. ca. 80 % Genauigkeit) nicht übertroffen werden. Jedoch können leicht verbesserte Ergebnisse mit einem stark angepassten Vision Transformer erreicht werden (circa 94 %) [42, @10].

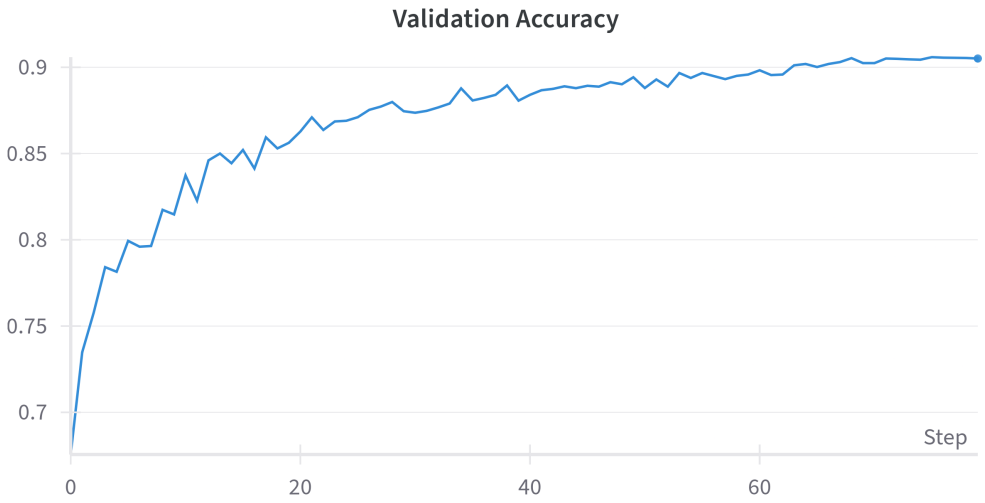


Abb. 8.1.: Genauigkeit auf dem Validation-Set des angepassten MAE

8.2 Schlussfolgerung

Das Pretraining mithilfe eines Masked Autoencoders zeigt Vorteile hinsichtlich der Performance der Klassifikation auf dem CIFAR-10-Datensatz. Es werden wesentlich schneller bessere Ergebnisse erreicht, welche mit derselben Transformer-Struktur ohne Pretraining nicht möglich wären. Des Weiteren verringert es den Effekt des Overfittings. Für domänenspezifische Daten, bei denen es meist nicht viele gelabelte Bilder gibt, kann dies sehr sinnvoll sein. Der Fakt, dass nur mit stark angepassten Vision Transformern ein wesentlich besseres Ergebnis erzielt werden kann, zeigt, wie gut der Masked Autoencoder zum Vortrainieren auf kleinen Datensets geeignet ist.

Aber es hat sich herausgestellt, dass die Struktur zu komplex für das kleine Datenset sein kann. Gerade die Transformer-Architektur.

9 SimMIM

SimMIM [44] ist ein leichtgewichtiges Framework, zumindest in Bezug gesetzt zum Masked Autoencoder. Die grundlegende Funktionsweise ist jedoch ähnlich. Es wird standardmäßig auch eine Transformer-Architektur als Encoder verwendet und die Bilder werden ebenfalls zufällig maskiert; dabei werden aber auch die maskierten Patches in den Encoder mit übergeben. Der Hauptunterschied liegt im fehlenden Decoder, denn der SimMIM verwendet lediglich einen simplen linearen Kopf, welcher direkt die RGB-Werte der maskierten Patches rekonstruiert. Es entsteht somit eine pixelweise Regression der fehlenden Bildanteile. Die Komplexität ist also wesentlich geringer und womöglich damit eher geeignet für kleine Datensets. Dies möchte ich nun untersuchen. [44]

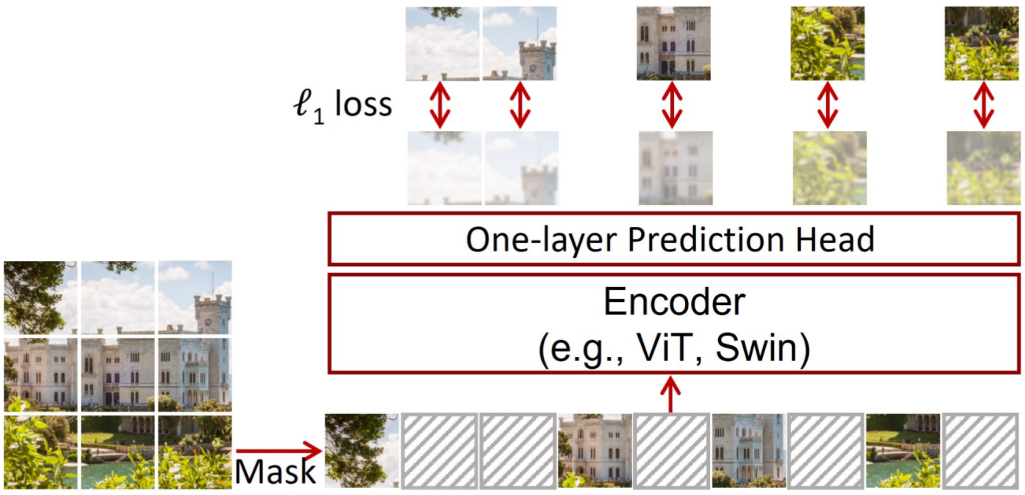


Abb. 9.1.: SimMIM Framework Abbildung, welches auf Pixelebene arbeiten mit Mean Absolute Error funktioniert ($l_1 - loss$). Zu sehen ist der Encoder und der Projektionskopf. Übernommen von Xie et al. [44].

9.1 Vorgehensweise und Rekonstruktion

Zuerst habe ich das Modell programmiert. Dafür habe ich zwei Quellen verwendet. Einmal das ursprüngliche Paper: “SimMIM: a simple Framework

for Masked Image Modeling” von Xie et al. [44] und ein GitHub-Repository von Bob McDear [@30]. Nachdem das Modell grundlegend funktioniert hat, habe ich versucht, den Encoder möglichst dem vom Masked Autoencoder anzupassen, um so eine faire Vergleichsbasis zu haben. Die Trainingsdauer habe ich auch wieder auf 100 Epochen gesetzt.

Die Rekonstruktionen 9.2, welche ich nach dem Pretraining bekommen habe, sehen nicht so gut aus, wie die eines Masked Autoencoders. Aber mit Blick auf den fehlenden Decoder doch akzeptabel.

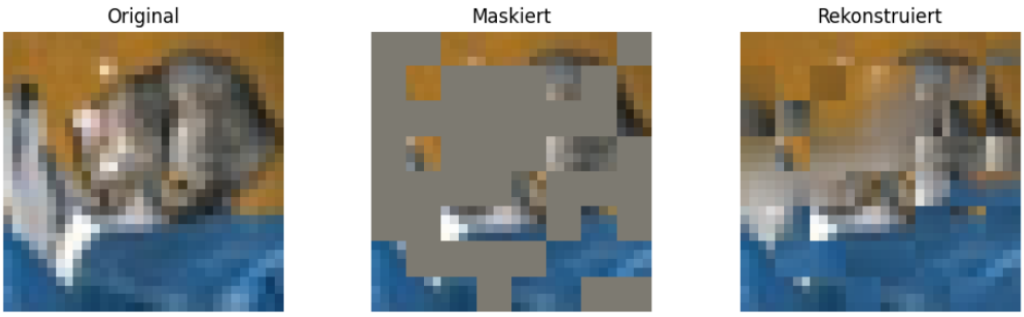


Abb. 9.2.: Rekonstruktion eines Bildes durch SimMIM

9.2 Ergebnisse und Interpretation

Wird der vortrainierte Encoder nun auf die Klassifikation trainiert, erhält man nach 100 Epochen Training eine Top-Accuracy von 90,35 %. Ohne weitere Verbesserungen an dem Modell. Das deutet darauf hin, dass der Masked Autoencoder leichteren Modellen nicht auf CIFAR-10 und weiteren kleinen Datensets überlegen ist und stärkt meine Hypothese.

Ein Grund für die gute Leistung ist womöglich, dass der lineare Kopf das Erlernen von Artefakten, welche nicht benötigt werden, reduziert. Damit wird das Overfitting verringert (lediglich eine Vermutung, müsste weiter untersucht werden)

Der Vorteil von SimMIM ist die leicht zu verstehende Architektur und die verbesserte Laufzeit, welche durch das Fehlen des Decoders zustande kommt. Mit diesem Ergebnis des SimMIM-Modells muss man die komplexe Struktur des Masked Autoencoders, zumindest für kleinere Datensätze, überdenken.

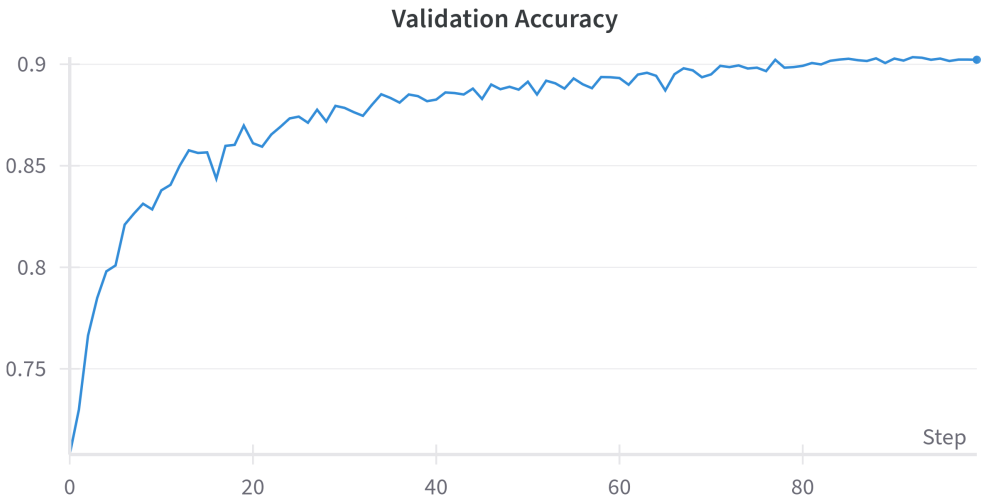


Abb. 9.3.: Performance des SimMIM-Modells auf CIFAR-10

10 Zusammenfassung und Ausblick

Abschließend lässt sich festhalten, dass Self-Supervised-Pretraining-Verfahren auf kleinen Datensets in der Bildklassifikation von Vorteil sind. Meine Verfahren haben keine Spitzenleistungen erzielt, aber zeigen deutlich, dass mit einem Pretraining durch Masked Image Modeling bessere Ergebnisse hinsichtlich der Generalisierung und der Genauigkeit erzielt werden können. Die Designprinzipien, welche sich in meiner Untersuchung über Masked Autoencoder herausgestellt haben, legen nahe, dass eine sorgfältige Auswahl dieser für den Erfolg der Ansätze essenziell ist.

Die experimentellen Ergebnisse zeigen eine Verbesserung der Klassifikationsgenauigkeit um 6 % durch die Optimierung des Pretraining-Verfahrens. Gegenüber einer Durchführung ohne Pretraining sogar eine Verbesserung um circa 11 % bei gleicher Trainingszeit. Klar wird, dass architektonische Entscheidungen für den Erfolg des Masked Autoencoders entscheidend sind. So ist eine Maskierungsrate von 75-85 % mit einer Patch-Größe von vier und einer Decoder-Tiefe von 96 mit dynamischer Token-Interaktion, meinen Untersuchungen nach, optimal.

Mit dem Kapitel 7 zeige ich, dass unterschiedliche latente Repräsentationen, basierend auf der Maskierungsrate, gelernt werden können. Dabei werden High-Level-Repräsentationen des Bildes erlernt, was durch die Minimierung des MSE, als auch durch die Maximierung des SSIM während des Lernprozesses gezeigt wurde. Des Weiteren wurde eine Verbindung zwischen dem MAE und kontrastivem Lernen gezogen und verdeutlicht, dass diese beiden Lernverfahren im Grunde dasselbe tun: (Positive) Bildpaare ausrichten.

Meine Arbeit stellt auch die Komplexität des Masked Autoencoders für kleine Datensätze in Frage. So habe ich am Ende der Arbeit gezeigt (Kapitel 9), dass auch wesentlich leichtere Modelle mindestens genauso gut auf den Datensätzen abschneiden können.

Zukünftige Arbeiten könnten sich darauf konzentrieren, etwas größere Datensets zu untersuchen. Etwas zwischen CIFAR-10 und ImageNet. Ebenso

wäre eine Analyse zu weiteren Downstream-Tasks, wie Anomalie-Detektion, interessant. Aber auch die offenen Fragen, welche ich in dieser Arbeit nicht beantworten konnte (z. B.: Grund für die große Patch-Size), haben ebenfalls Potenzial für weiterführende Forschungsarbeiten. Die Performance auf kleinen Datensets mit einem vorgefertigten Modell, welches auf ImageNet trainiert wurde, ist ebenfalls noch interessant, mit meinen Ergebnissen zu vergleichen. Insbesondere bei spezialisierten Klassifikationsaufgaben. Noch ein weiteres relevantes Thema ist, eine spezielle Vereinfachung von MAE für kleine Datensets zu finden oder SimMIM genauer auf kleine Datensets anzupassen.

Schlussendlich kann man sagen, dass Self-Supervised-Pretraining-Verfahren ein erfolgreiches Forschungsfeld ist, welches bereits in der Praxis angekommen ist. Meine Untersuchung zeigt, dass Pretraining mit Masked Image Modeling-Techniken auch auf kleinen Datensets zu großen Verbesserungen führt, indem es die Generalisierungsfähigkeiten stärkt und semantisch gute Merkmale erlernt. Damit kann es ebenfalls in der Praxis angewendet werden.

Literatur

[1]Mirza Samad Ahmed Baig, Syeda Anshrah Gillani, Abdul Akbar Khan und Shahid Munir Shah. *AttentionDrop: A Novel Regularization Method for Transformer Models*. arXiv:2504.12088 [cs]. 2025 (zitiert auf Seite 35).

[2]Hangbo Bao, Li Dong, Songhao Piao und Furu Wei. *BEiT: BERT Pre-Training of Image Transformers*. 2022. arXiv: 2106.08254 [cs.CV] (zitiert auf Seite 14).

[3]Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai und Aäron van den Oord. *Are we done with ImageNet?* 2020. arXiv: 2006.07159 [cs.CV] (zitiert auf Seite 43).

[8]Ting Chen, Simon Kornblith, Mohammad Norouzi und Geoffrey E. Hinton. "A Simple Framework for Contrastive Learning of Visual Representations". In: *ArXiv* abs/2002.05709 (2020) (zitiert auf den Seiten 8, 9).

[9]Yabo Chen, Yuchen Liu, Dongsheng Jiang et al. *SdAE: Self-distillated Masked Autoencoder*. 2022. arXiv: 2208.00449 [cs.CV] (zitiert auf den Seiten 14, 15).

[13]Jia Deng, Wei Dong, Richard Socher et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, S. 248–255 (zitiert auf den Seiten 2, 43).

[14]Jacob Devlin, Ming-Wei Chang, Kenton Lee und Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 24. Mai 2019. arXiv: 1810.04805 [cs]. Vorveröffentlichung (zitiert auf den Seiten 1, 11, 14, 21, 23).

[15]Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov et al. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 3. Juni 2021. arXiv: 2010.11929 [cs]. Vorveröffentlichung (zitiert auf den Seiten 1, 2, 5, 12–14, 17).

[17]Kaiming He, Xinlei Chen, Saining Xie et al. *Masked Autoencoders Are Scalable Vision Learners*. Version 3. 2021. Vorveröffentlichung (zitiert auf den Seiten 1, 2, 5, 12, 15, 17–19, 23, 24, 30, 32, 39, 45, 46).

[18]Vlad Hondru, Florinel Alin Croitoru, Shervin Minaee, Radu Tudor Ionescu und Nicu Sebe. *Masked Image Modeling: A Survey*. arXiv:2408.06687 [cs]. Juli 2025 (zitiert auf Seite 15).

[19]Zhicheng Huang, Xiaojie Jin, Chengze Lu et al. "Contrastive Masked Autoencoders are Stronger Vision Learners". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.4 (Apr. 2024), S. 2506–2517 (zitiert auf Seite 15).

[21]Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee und Fillia Makedon. *A Survey on Contrastive Self-supervised Learning*. 2021. arXiv: 2011.00362 [cs.CV] (zitiert auf Seite 5).

[23]Daniel Jurafsky und James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released January 12, 2025. 2025 (zitiert auf Seite 1).

[24]Gitit Kehat und James Pustejovsky. *Annotation Methodologies for Vision and Language Dataset Creation*. 10. Juli 2016. arXiv: 1607.02769 [cs]. Vorveröffentlichung (zitiert auf den Seiten 1, 5).

[25]Lingjing Kong, Martin Q. Ma, Guangyi Chen et al. *Understanding Masked Autoencoders via Hierarchical Latent Variable Models*. arXiv:2306.04898 [cs]. Juni 2023 (zitiert auf den Seiten 2, 45–48).

[26]Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto, 2009 (zitiert auf Seite 2).

[28]Yann LeCun, Yoshua Bengio und Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), S. 436–444 (zitiert auf Seite 12).

[29]Ilya Loshchilov und Frank Hutter. *Decoupled Weight Decay Regularization*. 4. Jan. 2019. arXiv: 1711.05101 [cs]. Vorveröffentlichung (zitiert auf Seite 19).

[31]Umberto Michelucci. *An Introduction to Autoencoders*. 11. Jan. 2022. arXiv: 2201.03898 [cs]. Vorveröffentlichung (zitiert auf den Seiten 6, 7).

[32]Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell und Alexei A. Efros. "Context Encoders: Feature Learning by Inpainting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, S. 2536–2544 (zitiert auf Seite 14).

[34]Christopher J. Shallue, Jaehoon Lee, Joseph Antognini et al. *Measuring the Effects of Data Parallelism on Neural Network Training*. arXiv:1811.03600 [cs]. Juli 2019 (zitiert auf Seite 19).

[37]Koustuv Sinha, Robin Jia, Dieuwke Hupkes et al. *Masked Language Modeling and the Distributional Hypothesis: Order Word Matters Pre-training for Little*. arXiv:2104.06644 [cs]. Sep. 2021 (zitiert auf Seite 1).

[39]Pascal Vincent, Hugo Larochelle, Yoshua Bengio und Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning - ICML '08*. Helsinki, Finland: ACM Press, 2008, S. 1096–1103 (zitiert auf den Seiten 13–15).

[41]Ross Wightman. *PyTorch Image Models*. 2019 (zitiert auf Seite 78).

[42]Gent Wu. *Powerful Design of Small Vision Transformer on CIFAR10*. arXiv:2501.06220 [cs]. Jan. 2025 (zitiert auf den Seiten 3, 55).

[44]Zhenda Xie, Zheng Zhang, Yue Cao et al. *SimMIM: A Simple Framework for Masked Image Modeling*. arXiv:2111.09886 [cs]. Apr. 2022 (zitiert auf den Seiten 3, 29, 30, 57, 58).

[45]Haohang Xu, Shuangrui Ding, Manqi Zhao und Dongsheng Jiang. *Masked Autoencoders are Robust Data Augmentors*. arXiv:2206.04846 [cs]. Apr. 2025 (zitiert auf den Seiten 3, 17, 18, 38).

[46]Ruijia Xu, Yixiao Ge, Kun Yi et al. *A Robustly and Effectively Optimized Pretraining Approach for Masked Autoencoder*. `https://openreview.net/forum?id=LHBiPX5BOwZ`. 2023 (zitiert auf den Seiten 2, 8, 14–16, 35, 36).

[47]Wu Zeng. "Image data augmentation techniques based on deep learning: A survey". In: *Mathematical Biosciences and Engineering* 21.6 (2024), S. 6190–6224 (zitiert auf den Seiten 37, 38).

[48]Chaoning Zhang, Chenshuang Zhang, Junha Song et al. *A Survey on Masked Autoencoder for Self-supervised Learning in Vision and Beyond*. arXiv:2208.00173 [cs]. Juli 2022 (zitiert auf den Seiten 18, 43).

[49]Qi Zhang, Yifei Wang und Yisen Wang. *How Mask Matters: Towards Theoretical Understandings of Masked Autoencoders*. arXiv:2210.08344 [cs]. März 2023 (zitiert auf den Seiten 2, 16, 45, 49–53).

[50]Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei und Ming Zhou. *Scheduled DropHead: A Regularization Method for Transformer Models*. arXiv:2004.13342 [cs]. Nov. 2020 (zitiert auf Seite 35).

[51]Zhou Wang, A.C. Bovik, H.R. Sheikh und E.P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* 13.4 (Apr. 2004), S. 600–612 (zitiert auf Seite 40).

[52]Haoran Zhu, Boyuan Chen und Carter Yang. *Understanding Why ViT Trains Badly on Small Datasets: An Intuitive Perspective*. arXiv:2302.03751 [cs]. Feb. 2023 (zitiert auf Seite 43).

Webpages

[@4]Jonathan Bgn. *The Rise of Self-Supervised Learning*. Letzter Zugriff: 29.07.2025. 31. Dez. 2020. URL: `https://jonathanbgn.com/2020/12/31/self-supervised-learning.html` (zitiert auf den Seiten 5, 6).

[@5]Zach Bobbitt. *How to Conduct Levene's Test for Equality of Variances in R*. Zugegriffen am 24. Juni 2025. 2022. URL: `https://www.statology.org/levenes-test-r/` (zitiert auf Seite 26).

[@6]Zach Bobbitt. *MAE vs. RMSE: Which Metric Should You Use?* Zugegriffen am 03. Juni 2025. 2021. URL: `https://www.statology.org/mae-vs-rmse/` (zitiert auf Seite 39).

[@7]Bohsun Chen. *Understanding Huber Loss function: Insights from Applications*. Zugegriffen am 03. Juni 2025. 2024. URL: %5Curl%7Bhttps://medium.com/@devcharlie2698619/understanding-huber-loss-function-insights-from-applications-5c1c5145d2c4%7D (zitiert auf den Seiten 39, 40).

[@10]Cydia2018. *ViT-cifar10-pruning: Vision Transformer Pruning*. Zugriff am 7. August 2025. 2021. URL: https://github.com/Cydia2018/ViT-cifar10-pruning (zitiert auf den Seiten 3, 55).

[@11]Ingo Dachwitz. *Data Workers' Inquiry: Die versteckten Arbeitskräfte hinter der KI erzählen ihre Geschichten*. Zugegriffen am 23. Mai 2025. Netzpolitik.org. 2024. URL: https://netzpolitik.org/2024/data-workers-inquiry-die-versteckten-arbeitskraefte-hinter-der-ki-erzaehlen-ihre-geschichten/ (zitiert auf Seite 1).

[@12]DataScientest. *Convolutional Neural Network (CNN): Alles, was Du wissen solltest*. Zugegriffen am 15. Juli 2025. DataScientest. Juli 2025. URL: https://datascientest.com/de/convolutional-neural-network-2 (besucht am 15. Juli 2025) (zitiert auf Seite 13).

[@16]GeeksforGeeks. *Vision Transformer (ViT) Architecture*. (Last updated). Zugegriffen: 15.Juli2025. Jan. 2025. URL: https://www.geeksforgeeks.org/deep-learning/vision-transformer-vit-architecture/ (besucht am 15. Juli 2025) (zitiert auf den Seiten 12, 13).

[@20]IcarusWizard. *MAE: Implementation of Masked Autoencoders*. Zugegriffen am 04. Juni 2025. 2022. URL: https://github.com/IcarusWizard/MAE (zitiert auf Seite 17).

[@22]Sonam Jindal. *Valuing Data Enrichment Workers: The Case for a Human-Centric Approach to AI Development*. Zugegriffen am 23. Mai 2025. United Nations Chronicle. 2023. URL: https://www.un.org/en/un-chronicle/valuing-data-enrichment-workers-case-human-centric-approach-ai-development (zitiert auf Seite 1).

[@27]Niklas Lang. *Understanding Mean Squared Error (MSE) – A Key Metric in Data Analysis!* Zugegriffen am 05. Juni 2025. 2023. URL: https://databasecamp.de/en/statistics/mean-squared-error (zitiert auf Seite 17).

[@30]Bob McDear. *SimMIM PyTorch: model.py*. Accessed: 2025-07-17. 2023. URL: https://github.com/BobMcDear/simmim-pytorch/blob/main/model.py (zitiert auf Seite 58).

[@33]Ruian7P. *Tiny-ViT-from-Scratch-on-CIFAR100: Training Vision Transformers from Scratch on CIFAR-100*. Zuletzt abgerufen am 12. August 2025. 2025. URL: https://github.com/Ruian7P/Tiny-ViT-from-Scratch-on-CIFAR100 (zitiert auf Seite 78).

[@35]Shubham. *Huber Loss – Loss function to use in Regression when dealing with Outliers*. Zugegriffen am 03. Juni 2025. 2025. URL: https://mlexplained.blog/2023/07/31/huber-loss-loss-function-to-use-in-regression-when-dealing-with-outliers/ (zitiert auf Seite 40).

[@36]Sumit Singh. *BERT Explained: State-of-the-art Language Model For NLP*. Zugegriffen am 01. Juni 2025. 2023. URL: https://www.labellerr.com/blog/bert-explained-state-of-the-art-language-model-for-nlp/ (zitiert auf den Seiten 1, 11, 12).

[@38]Arunn Thevapalan. *ANOVA-Test: Ein detaillierter Leitfaden mit Beispielen*. Zugegriffen am 24. Juni 2025. 2024. URL: https://www.datacamp.com/de/tutorial/anova-test (zitiert auf den Seiten 25, 26).

[@40]Ross Wightman. *PyTorch Image Models*. 2019. URL: %5Curl%7Bhttps://github.com/huggingface/pytorch-image-models%7D (zitiert auf Seite 18).

[@43]Laurenz Wuttke. *Was ist Supervised Learning (Überwachtes Lernen)?* Letzter Zugriff: 29.07.2025. 10. Juli 2022. URL: https://datasolut.com/wiki/supervised-learning/ (zitiert auf Seite 5).

Abbildungsverzeichnis

2.1. Struktur eines Autoencoders. Zu sehen ist der Input (Bild mit einer 4), welches in den Encoder übergeben wird. Der Encoder wandelt den Input in die latente Repräsentation um, welche in der Mitte als vier Rechtecke abgebildet ist. Der Decoder rekonstruiert nun das Bild wieder aus dem latenten Raum. Abbildung wurde leicht angepasst von Michelucci [31] übernommen 6

2.2. Schematische Darstellung eines kontrastiven Lernframeworks, mit einzelnen Schlüsselkomponenten (1-4). Aus dem Input x werden zwei Abbildungen durch die Augmentation erzeugt. Der Encoder projiziert diese dann in den latenten Raum, in dem dann durch den Projektionskopf und die Verlustfunktion die Übereinstimmung maximiert werden soll. Dies führt zur Entstehung eines initialen positiven Paars im Merkmalsraum. Abbildung in abgeänderter Form übernommen von Chen et al. [8]. 9

3.1. BERT Pre-Training Architektur. In dem Satz, welcher aus fünf Wörtern besteht, wird das vierte Wort maskiert, das Modell soll dann das maskierte Wort wieder richtig hervorsagen. Abbildung übernommen von Sumit Singh [@36] 12

3.2. Architektur eines Vision Transformers; Abbildung übernommen von Dosovitskiy et al. [15] 13

4.1. Architektur eines Masked Autoencoders; Abbildung übernommen von He et al. [17] 18

4.2. Trainingsverlauf über 300 Epochen / 600 Trainingsteps 20

4.3. Rekonstruktion nach Training 21

4.4. Genauigkeit auf dem Validation-Set, einmal mit und einmal ohne Pretraining. x-Achse: Anzahl der Epochen; y-Achse: Genauigkeit . . 22

5.1. Rekonstruktion nach Training 24

5.2. Accuracy mit jeweiliger Mask-Ratio und Strategie 30

5.3. Genauigkeit bei entsprechender Embedding-Dimension 34

7.1. Hierarchischer Graph eines Bildes mit Pixeln $x_1 - x_{11}$ und den lernbaren latenten Variablen $z_1 - z_{10}$. Dabei sind die oberen latenten Variablen die High-Level-Repräsentationen. Bild von Kong et al. [25] nachempfunden 46
7.2. Unterschiedliche Maskierungsraten (aufsteigend von links) und deren gelernte latente Repräsentation welche blau markiert ist. Bild von Kong et al. [25] nachempfunden 46
7.3. Rekonstruktionsvalidierung; Ergebnisse des Experiments, welche Repräsentationen gelernt werden. 48
7.4. Die Abbildung zeigt den bipartiten Maskierungs Graph und den Augmentations Graphen. Leicht abgeändert von [49] übernommen 51

8.1. Genauigkeit auf dem Validation-Set des angepassten MAE 56

9.1. SimMIM Framework Abbildung, welches auf Pixelebene arbeiten mit Mean Absolute Error funktioniert ($l_1 - loss$). Zu sehen ist der Encoder und der Projektionskopf. Übernommen von Xie et al. [44]. 57
9.2. Rekonstruktion eines Bildes durch SimMIM 58
9.3. Performance des SimMIM-Modells auf CIFAR-10 59

A.1. Drei unterschiedliche Maskierungsstrategien mit maskiertem Bild, rekonstruierten Bild und Originalbild. 76
A.2. Performance meines MAEs auf CIFAR-100 79

Tabellenverzeichnis

5.1. Ergebnisse aller Durchläufe pro Maskierungsrate 25
5.2. Ergebnisse des Shapiro-Wilk-Test für jede Gruppe 26
5.3. Durchschnitt und Standardabweichung der Genauigkeit 28
5.4. Best erreichte Genauigkeit für jede Strategie 30
5.5. Genauigkeit je Patch-Größe 32
5.6. Accuracy mit Dropout und ohne Dropout im Decoder 36
5.7. Ergebnisse der Untersuchung zur Data Augmentation 38
5.8. Ergebnisse der Untersuchung zur Data Augmentation 41

8.1. Parameter des optimierten Durchlaufs 55

A.1. Trainingsverläufe und alle Rekonstruktionen 77
A.2. Trainingsverläufe und alle Rekonstruktionen Fortführung 78

Programmausdrucksverzeichnis

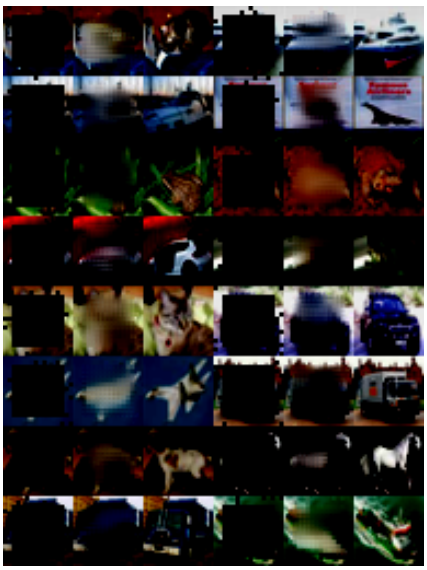
5.1. Shapiro-Wilk-Test 25
5.2. Levene-Test 26
5.3. Attention-Layer mit Dropout im Decoder 36

Appendix

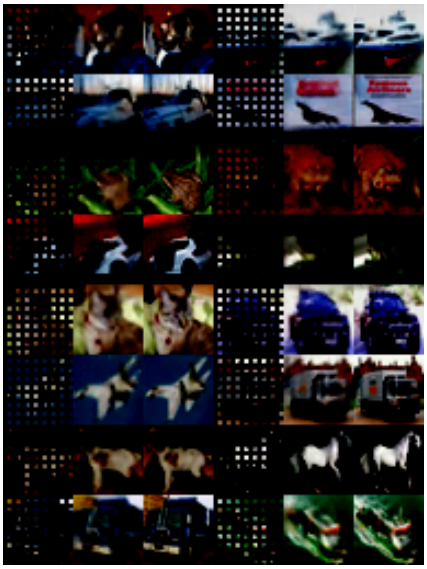
A

A.1 Unterschiedliche Maskierungsstrategien und deren Rekonstruktion

(a) zufällige Maskierung (85 %)



(b) blockweise Maskierung (60 %)



(c) Raster-Maskierung (75 %)

Abb. A.1.: Drei unterschiedliche Maskierungsstrategien mit maskiertem Bild, rekonstruierten Bild und Originalbild.

A.2 Ablauf aller Untersuchungen

In folgender Tabelle wird der Trainingsverlauf vieler Untersuchungen verlinkt. Darunter zählt sowohl das Pretraining des Masked Autoencoders als auch das Fine-Tuning zur Klassifikation. Die Links führen zu meinen Projekten auf der Website: ”Weights & Biases”.

Anmerkung: Teilweise können die Metadaten nicht stimmen, da ich sie vorübergehend falsch geloggt habe. Darüber hinaus können die Runtimes von manchen Durchläufen nur 3 Sekunden betragen, diese wurden dann aus einem anderen Projekt importiert / verschoben.

Tab. A.1.: Trainingsverläufe und alle Rekonstruktionen

Untersuchung	**Link**
Maskierungsraten – Pretraining	`https://wandb.ai/warmund-jgu/MAE_Cifar10_mask_ratios_v3?nw=nwuserwarmund`
Maskratios – Klassifikation	`https://wandb.ai/warmund-jgu/vit-mae-comparison-mask-ratios?nw=nwuserwarmund`
Maskierungsarten – Autoencoder	`https://wandb.ai/warmund-jgu/Masking_Strategy_v2?nw=nwuserwarmund`
Maskierungsarten – Klassifikation	`https://wandb.ai/warmund-jgu/vit-mae-comparison-mask-ratios?nw=nwuserwarmund`
Decoder Tiefe – Autoencoder	`https://wandb.ai/warmund-jgu/Decoder_Depth?nw=nwuserwarmund`
Decoder Tiefe – Klassifikation	`https://wandb.ai/warmund-jgu/vit-mae-comparison-depth?nw=nwuserwarmund`
Embedding Dim. – Autoencoder	`https://wandb.ai/warmund-jgu/Decoder_EMBED_DIM/workspace?nw=nwuserwarmund`
Embedding Dim. – Klassifikation	`https://wandb.ai/warmund-jgu/vit-mae-comparison-embed-dim?nw=nwuserwarmund`

Tab. A.2.: Trainingsverläufe und alle Rekonstruktionen Fortführung

Untersuchung	Link
Dropout – Klassifikation	`https://wandb.ai/warmund-jgu/vit-mae-comparison-dynamic?nw=nwuserwarmund`
Patch-Größe – Klassifikation	`https://wandb.ai/warmund-jgu/vit_mae_patch_size_comp?nw=nwuserwarmund`

A.3 Weitere nicht aufgelistete Untersuchungen

Ich habe noch weitere nicht aufgelistete Untersuchungen durchgeführt, welche jedoch nicht in die Arbeit aufgenommen wurden, da sie aufgrund ihres Entwicklungsstands noch keine belastbaren Aussagen ermöglichen.

A.3.1 ImageNet Pretraining

Ich habe einen vortrainierten Masked Autoencoder aus der Bibliothek Timm geladen [41] und diesen für die Klassifikation von CIFAR-10 feinjustiert. Mein Ergebnis war eine 94,54 % Genauigkeit. Damit hat er ein besseres Ergebnis erzielt als der auf CIFAR-10 vortrainierte Masked Autoencoder.

Das Ergebnis kann aber, wie schon in meiner Zusammenfassung gesagt, bei stark spezialisierten Datensets nochmal komplett anders aussehen.

A.3.2 CIFAR-100

Zusätzlich wollte ich noch meine Architektur auf etwas größeren Datensets evaluieren. Dabei hat sich herausgestellt, dass meine Architektur dafür nicht geeignet ist. Die Ergebnisse zeigen nach 80 Epochen nur eine Genauigkeit von knapp über 60 %, was keine Verbesserung gegenüber ViT-Tiny-Modellen aufweist [@33].

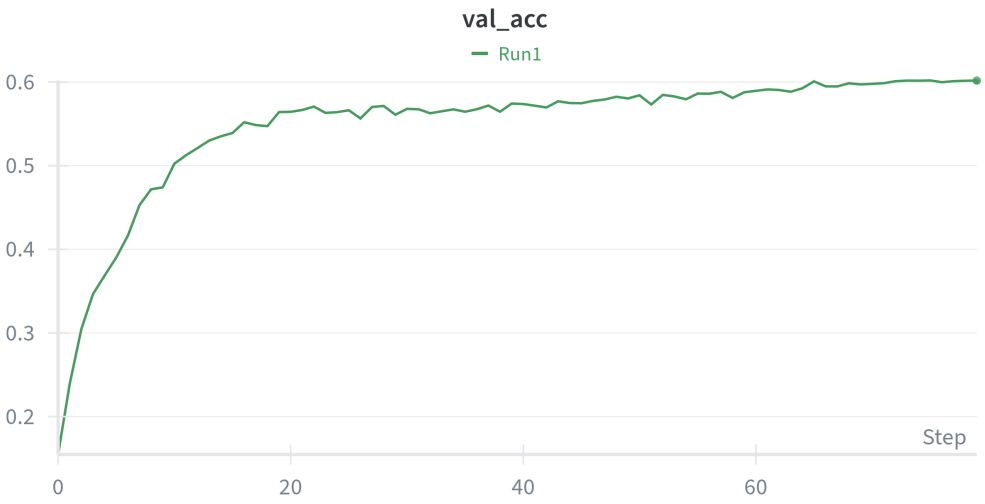


Abb. A.2.: Performance meines MAEs auf CIFAR-100

Kolophon

Diese Arbeit wurde mit LaTeX 2_ε gesetzt. Sie verwendet den von Ricardo Langner entwickelten Stil *Clean Thesis*. Das Design des Stils *Clean Thesis* ist von Benutzerhandbüchern von Apple Inc. inspiriert.

Laden Sie den Stil *Clean Thesis* unter `http://cleanthesis.der-ric.de/` herunter.

Anpassungen an den Stil des Instituts für Informatik sind unter `https://gitlab.rlp.net/institut-fur-informatik/cleanthesis-jgu` zu finden.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe (dazu zählen auch „KI-Werkzeuge“)). Sämtliche wörtlichen und sinngemäßen Übernahmen und Zitate sind kenntlich gemacht und nachgewiesen. Auch beim Einsatz von KI-Werkzeugen[1] stellt die Arbeit klar dar, in welchem Umfang diese genutzt wurden und welche Inhalte dadurch erzeugt oder beeinflusst wurden[2]. Ich versichere, dass ich keine Hilfsmittel verwendet habe, deren Nutzung die Prüferinnen und Prüfer explizit ausgeschlossen haben.

Mit Abgabe der vorliegenden Leistung übernehme ich die Verantwortung für das eingereichte Gesamtprodukt. Ich verantworte damit auch alle KI-generierten Inhalte, die ich in meine Arbeit übernommen habe. Die Richtigkeit übernommener (KI-generierter) Aussagen und Inhalte habe ich nach bestem Wissen und Gewissen geprüft.

Ich habe die Arbeit nicht zum Erwerb eines anderen Leistungsnachweises in gleicher oder ähnlicher Form eingereicht. Von der Ordnung zur Sicherung guter wissenschaftlicher Praxis und zum Umgang mit wissenschaftlichem Fehlverhalten habe ich Kenntnis genommen.

Mir ist bekannt, dass ein Verstoß gegen die genannten Punkte prüfungsrechtliche Konsequenzen hat und insbesondere dazu führen kann, dass die Studien- und Prüfungsleistung als mit „nicht bestanden“ bewertet wird. Die Einschreibung kann für bis zu zwei Jahre widerrufen werden, wenn Studierende

[1] Mit „KI-Werkzeugen“ werden computergestützte Systeme bezeichnet, die auf Basis maschinellen Lernens neue Inhalte generieren können (z. B. ChatGPT, Gemini, Claude, LLaMA, Midjourney, Stable Diffusion o. ä.).

[2] Die Nutzung von KI-Werkzeugen muss dann kenntlich gemacht werden, wenn Sie den Kern der Aufgabenstellung in von den Prüfern nicht anzunehmender Weise berührt: Sollte nichts anderes explizit mit den Prüferinnen und Prüfern vereinbart worden sein, so ist dies der Fall, sobald wesentliche Teile der eingereichten Arbeit (z. B. ganze Sätze des Textes, ganze Abbildungen, nicht-offensichtliche Teile von Rechnungen oder Beweisen, mehrere Zeilen von Programmcode am Stück) von solchen Werkzeugen generiert wurden und wörtlich oder in abgewandelter Form in die Arbeit übernommen wurden oder wesentliche Konzepte oder Ideen (z. B. eine Gliederung der Literatur oder ein Lösungsansatz für ein Problem) von KI-Werkzeugen generiert wurden.

zweimal oder häufiger bei Prüfungsleistungen täuschen (§ 69 Abs. 4 und 5 HochSchG).

Mainz, 12. August 2025

Patrick Radau

Nutzung von KI-Tools.

KI Tool	Genutzt für	Grund	Wann
Perplexity	Satzbau und Kommasetzung ohne den Inhalt zu verändern.	Rechtschreibüberprüfung	dauerhaft
ChatGPT	Formatierung von Bildern in LaTeX	Fehlerbehebung	Bei Abbildung A.1