„Customized Object Detection mit Transfer Learning am Beispiel von Bilddaten des Badischen Landesmuseums“

vorgelegt von

Oliver Gorczyca, Anas Arodake

Thesis für angestrebten akademischen Grad

Bachelor of Science (B.Sc.)

Kiel, 31.01.2023

Fachbereich	Informatik und Elektrotechnik
Studiengang	Informationstechnologie
Matrikelnummer	935160, 935109
Erstgutachter	Prof. Dr. Stephan Schneider
Zweitgutachter	Sonja Thiel

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Erkennung und Lokalisierung von Minderjährigen auf digitalen Bildern, die von dem Badischem Museum in Karlsruhe zur Verfügung gestellt wurden. Die Sammlung historischer Aufnahmen enthält einige Bilder von Kindern, die aus Gründen des Schutzes vor Veröffentlichung ausgeschlossen werden müssen. Ziel des Projekts war es, sämtliche Kinderbilder zu identifizieren und dem Museum zu melden.
Zur Durchführung des Projekts wurde im ersten Schritt eine Vorab-Suche mittels eines pre-trained model zur Erkennung von Menschenbildern durchgeführt. Dabei konnte die Klasse 'person' auf Bildern bereits erfolgreich detektiert werden, wodurch ein manuelles Durchsuchen und Trennen der Bilder vermieden werden konnte. Die resultierenden Bilder wurden anschließend erneut mit dem gleichen pre-trained model, CenterNet Zhou et al. (2019), pre-annotiert und mit gut platzierten Bounding Boxen versehen. Diese Bilder dienten als Grundlage für das weitere Training eines eigenen Object Detection Modells, das in der Lage ist, Kinder zu erkennen.
Die vorliegende Arbeit verfolgt nicht nur das Ziel, Kinder auf Museumsbildern zu identifizieren, sondern bildet auch gleichzeitig eine Grundlage zur Bekämpfung von Kinderpornografie. Diese Zielsetzung war von Anfang an Teil dieser Arbeit. Die Wahl von CenterNet scheint deswegen sehr passend, da dieses Modell auch menschliche Positionen erkennen kann und somit einen optimalen Ansatz für die automatisierte Identifikation von kinderpornografischem Material bietet.
Die vorgestellte Methode zeigt, dass durch den Einsatz moderner Technologien wie CenterNet eine effiziente und automatisierte Erkennung von Kindern auf digitalen Bildern möglich ist und leistet damit einen wichtigen Beitrag zur Prävention von Kindesmissbrauch.

Danksagung

im Rahmen unserer Arbeit zum Thema Bilderkennung möchten wir uns herzlich bei allen Personen bedanken, die uns während der Durchführung des Projektes unterstützt haben.
Unser besonderer Dank gilt Prof. Dr. Stephan Schneider für seine herzliche Begleitung und seine wertvollen Ratschläge während der gesamten Projektphase. Sein Fachwissen und seine Erfahrung haben uns sehr geholfen, die Herausforderungen bei der Objekterkennung zu bewältigen.
Wir möchten auch Frau Sonja Thiel für ihre umfassende Unterstützung danken. Sie hat uns mit wichtigen Informationen und wertvollem Feedback versorgt, die uns dabei geholfen haben, unser Projekt erfolgreich abzuschließen.
Darüber hinaus möchten wir uns bei unseren Partnerinnen Martje Groeneveld und Yana Kolbach für ihre unermüdliche Motivation und ihren seelischen Beistand bedanken. Ohne ihre Unterstützung wäre die Fertigstellung des Projekts sicherlich viel schwieriger gewesen.
Herrn Dirk Jeß danken wir auch für das Probe lesen unserer Arbeit.
Ohne die Hilfe und Unterstützung dieser Personen wäre unser Projekt nicht möglich gewesen. Wir sind sehr dankbar für ihre wertvollen Beiträge und fühlen uns geehrt, solche großartigen Unterstützer in unserem Leben zu haben.

Arbeitsaufteilung

Diese Thesis wurde in gemeinsamer Arbeit erarbeitet.

Folgende Teile wurden von Anas Arodake erarbeitet:

Einleitung

- Problemstellung und Zielsetzung
- Auswahl der Methodik

Theoretischer Hintergrund

- Convolutional Neuronal Network - Layer in einem CNN
- Image Classification
- Object Detection

Praktische Anwendung des Systems

- Daten
- Transfarelearning und Modell
- Ergebnisse

Folgende Teile wurden von Oliver Gorczyca erarbeitet:

Einleitung

- Vorgehensweise
- Auswahl der Methodik

Theoretischer Hintergrund

- Convolutional Neuronal Network - Training von CNNs
- Convolutional Neuronal Network - Lernmethoden
- Transfer Learning

Praktische Anwendung des Systems

- Grundlagen
- Hardware- und Softwareumgebung
- Reflektion

Inhaltsverzeichnis

1 Einleitung

Auf den folgenden Seiten werden die Begriffe Modell und Netz synonym verwendet.

1.1 Problemstellung und Zielsetzung

Das Badische Museum in Karlsruhe besitzt große Sammlungen von Aufnahmen aus verschiedenen Quellen, die jedes Mal manuell durchsucht und sortiert werden müssen. Dies ist sehr zeitaufwendig und fehleranfällig für die beteiligten Personen. Zusätzlich enthält die Sammlung Bilder von Minderjährigen unterschiedlichen Alters, darunter auch leicht- / unbekleidete Kinder, die aus Datenschutzgründen von einer Veröffentlichung ausgeschlossen werden müssen, auch wenn die darauf abgebildeten Personen bereits verstorben sind.

Um dieses Problem zu beheben, ist eine maschinelle Lösung im Bereich der Computer Vision denkbar, die sich mit der digitalen Bildverarbeitung befasst. Allerdings ergeben sich hier einige Herausforderungen. So sind einige Machine Learning-Modelle zwar in der Lage, Kinder auf einem Bild zu erkennen und zu lokalisieren, aber das Problem dabei ist, dass diese Modelle Männer, Frauen und Kinder alle unter der Klasse 'person' einordnen.
Somit stellt sich die Frage: Wie kann die manuelle Verwaltung einer großen Bildersammlung durch Künstliche Intelligenz effektiv ersetzt werden, um Kinder auf Fotos zu klassifizieren, um so Zeit bzw. Ressourcen zu sparen und gleichzeitig sicher zu stellen, dass unangemessene Fotos vor der Veröffentlichung ausgeschlossen werden?

1.2 Vorgehensweise

Ein Lösungsansatz für das Problem bietet ein KI-gestütztes System. Das System kann in mehrere Schritte unterteilt werden:

Datenvorbereitung: Die Bilder müssen für die Verarbeitung durch das KI-System vorbereitet werden. Dazu gehört das Resizing der Bilder, das Entfernen von Metadaten und das Taggen von Bildern mit zusätzlichen Informationen, um die Klassifizierung zu unterstützen.

Training des KI-Modells: Ein KI-Modell muss trainiert werden, um Bilder automatisch zu klassifizieren und zu markieren. Das Modell kann auf Basis von Supervised Learning oder Unsupervised Learning trainiert werden, um die bestmöglichen Ergebnisse zu erzielen.

Integration in ein Workflow-System: Nachdem das KI-Modell trainiert wurde,ist es nötig, es in ein Workflow-System zu integriert. Dieses System kann eine Benutzeroberfläche für die manuelle Überprüfung der klassifizierten Bilder durch Mitarbeiter/-innen bereitstellen, um sicherzustellen, dass alle unangemessenen Fotos ausgeschlossen werden. Das System sollte auch Regeln für die automatische Ablehnung von bestimmten Bildern, basierend auf den Klassifikationsergebnissen, implementieren.

Kontinuierliche Verbesserung: Es ist wichtig, das KI-Modell und das Workflow-System regelmäßig zu überwachen und zu verbessern, um die Leistung zu optimieren und sicherzustellen, dass es den aktuellen Anforderungen entspricht.

Insgesamt bietet ein KI-gestütztes System eine effiziente und genaue Möglichkeit, eine große Bildersammlung zu durchsuchen, Objekte auf Fotos zu klassifizieren und kritische Fotos vor der Veröffentlichung auszuschließen. Der Einsatz von KI-Systemen kann dazu beitragen, Zeit und Ressourcen zu sparen und die Effizienz der Bildverarbeitung zu verbessern.

1.3 Auswahl der Methodik

Die Lösung des Problems der 'Kind-Erkennung' bietet verschiedene Optionen, von denen jede ihre Vor- und Nachteile aufweisen.

Methode 1: Image Segmentation: Bei der Image Segmentation werden alle Pixel in einem Bild mit einem Label versehen. Es können mehrere Labels in einem Bild enthalten sein. Das Problem bei dieser Methode ist der erhebliche Zeitaufwand beim Labeln der Bilder.

Methode 2: Unüberwachtes Lernen, Clustering: Man lässt ein Machine Learning Modell die unterschiedlichen Klassen der Bilder erkennen, um die Bilder vorzusortieren. Dadurch wird erhofft, bisher unentdecktes Wissen aus den Bildern zu extrahieren. Ein Problem dieser Methode ist der hohe Einarbeitungsaufwand und die Vielfalt der Klassen, die in den Bildern enthalten sein können.

Methode 3: Image Classification mit Transfer Learning: Bei dieser Methode wird ein bereits vortrainiertes Modell genutzt, das auf einer großen Datenmenge trainiert wurde. Das Modell wird dann auf die Erkennung von Kindern spezialisiert. Transfer Learning ist ein effektiver Ansatz, um ein Modell schnell und effizient an eine neue Aufgabe anzupassen. Der Vorteil dieser Methode ist, dass der Einarbeitungsaufwand im Vergleich zu Methode 2 geringer ist, und dass bereits vorhandene Wissen des vortrainierten Modells genutzt werden kann. Ein Nachteil ist jedoch, dass das Modell möglicherweise nicht optimal auf die spezifische Aufgabe der 'Kind-Erkennung' angepasst werden kann, da auf vielen Bildern mehrere Menschen gleichzeitig zu sehen sind.

Methode 4: Object Detection und Klassifikation: Bei der Object Detection werden Objekte auf den Bildern erkannt und lokalisiert. Anschließend kann das Modell die Kinder auf den Bildern erkennen. Der Vorteil dieser Methode ist, dass sie speziell auf die Aufgabe der 'Kind-Erkennung' zugeschnitten werden kann. Zudem können die Kinder-Bilder mit einer verpixelten Maskierung versehen werden, um ihre Identität zu schützen und sie gezielt und explizit von der Veröffentlichung auszuschließen. Ein Nachteil ist jedoch der höhere zeitaufwand beim Labeln der Daten.

Entscheidung: Nach Abwägung der Vor- und Nachteile jeder Methode fiel die Entscheidung auf die Methode Object Detection. Diese Methode lässt sich am besten an die spezifischen Anforderungen anpassen, die diese Arbeit zeigen wird.

2 Grundlagen Objekterkennung und Lokalisation

2.1 Convolutional Neuronal Network

Das Convolutional Neuronal Network (CNN), auch bekannt als faltungsneuronales Netz, ist eine spezielle Form des neuronalen Netzes, das sich durch seine hohe Genauigkeit bei der Merkmalerkennung von Bildern auszeichnet. Die Convolutional-Layer innerhalb des CNNs nutzen Filter, um spezifische Merkmale des Bildes zu identifizieren und zu extrahieren, beispielsweise Kanten, Formen oder Texturen. Das CNN nutzt eine Kombination aus verschiedenen Schichten, wie Convolutional-Layer, Pooling-Layer und Fully-Connected-Layer, um Merkmale aus den Eingabedaten zu extrahieren und eine Klassifikationsaufgabe zu bewältigen.

2.1.1 Layer in einem CNN

Convolutional-layer

Die Faltungsschicht ist das wichtigste Strukturelement eines CNN. Sie besteht aus mehreren Filtern, die Merkmale aus dem Eingangsbild extrahieren. Die Faltungsfilter bestehen aus einer rechteckigen Matrix von Gewichten (Kernel). In jedem Schritt wird die Summe der Produkte aus den Elementen des Kernel-Filters und den Pixelwerten im Fenster berechnet (sehe Abbild 2.1), um eine so genannte Aktivierungskarte (ein feature map oder convoluted image) zu erstellen. Mehrere Filter erzeugen mehrere Merkmalskarten, die verschiedene Merkmale im Ausgangsbild als zweidimensionale Matrix darstellen. Die Aktivierungskarte (feature map) wird an die nächste Schicht weitergegeben, wo nach dem gleichen Prinzip noch mehr Merkmale extrahiert werden können. Die Tiefe der Faltungsschichten kann bis zu mehreren Schichten betragen, was CNN hilft, komplexere Merkmale zu untersuchen. Albawi et al. (2017).

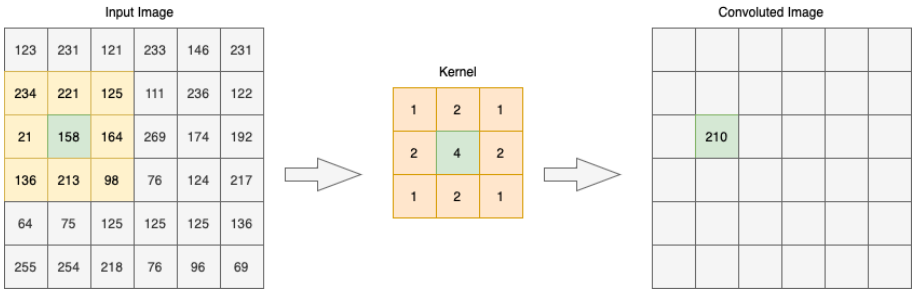

Abbildung 2.1: Convolution with kernel

Pooling-layer

Der Pooling-Layer wird in der Regel nach dem Convolutional-Layer in CNN eingesetzt, um die Komplexität der erzeugten Feature Maps zu reduzieren. Das Ziel dieses Layers besteht darin, die Anzahl der Parameter und die Rechenzeit beim Training des Netzes zu reduzieren. In der Regel wird die Max-Pooling-Operation in Pooling-Layern verwendet, bei der die Aktivierungskarte (Feature Map) in kleine Bereiche unterteilt wird (z. B. 2x2 oder 3x3 Pixel). Nur der maximale Wert in jedem Bereich wird in eine weitere 2D-Matrix übertragen. Der resultierende Output ist eine neue Aktivierungskarte, die etwa halb so groß ist wie die ursprüngliche Aktivierungskarte. Eine weitere wichtige Funktion des Pooling-Layers ist das Overfitting-Risiko zu reduzieren, indem eine Art von Regularisierung eingeführt wird. Durch das Entfernen von Informationen aus der Aktivierungskarte wird das Netzwerk gezwungen, nur die wichtigsten Merkmale zu extrahieren, was dazu beitragen kann, Überanpassung (overfitting) zu vermeiden.Jie and Wanda (2020)

Zwischen aufeinanderfolgenden Convolutional-Layern werden Pooling-Layer implementiert, deren Output als Input des darauf folgenden Convolutional-Layers fungiert. Nach mehreren solchen Schichten wird der Output des letzten Pooling-Layers an das Fully-Connected-Layer des Netzes weitergeleitet, welches die endgültige Klassifikation des Eingabebildes durchführt.

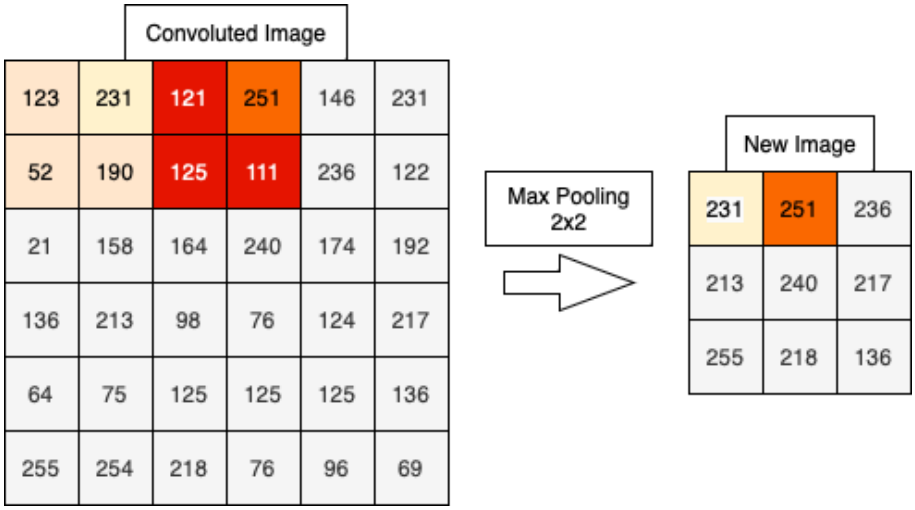

Abbildung 2.2: Max Pooling

Fully-Connected-layer

Das Fully-Connected-Layer in einem Convolutional Neural Network (CNN) besteht aus einer Schicht von Neuronen, die mit jedem Neuron in der vorherigen Schicht vollständig verbunden sind. Es ähnelt einem traditionellen neuronalen Netz, das für die Klassifikation oder Regressionsaufgaben verwendet wird. Die Aktivierungskarten, die aus den Convolutional- und Pooling-Layern erzeugt wurden, werden zu einem Vektor flachgedrückt (Flattening), um als Eingabe für das Fully-Connected-Layer zu dienen. Der Vektor enthält alle Merkmale des Eingabebildes, die von den vorherigen Schichten extrahiert wurden. Jedes Neuron im Fully-Connected-Layer berechnet eine gewichtete Summe der Aktivierungen der vorherigen Schicht und wendet dann eine Aktivierungsfunktion an, um einen Output zu erzeugen. Die Gewichte für jedes Neuron werden während des Trainings automatisch angepasst, um die

Leistung des Netzes auf einer Validierungsdatensatz zu verbessern. Das Fully-Connected-Layer am Ende des CNN ist für die Klassifikation oder Regressionsaufgabe verantwortlich. Für eine Klassifikation beispielsweise besteht der Ausgangslayer aus einer Schicht von Neuronen, die für jede Klasse eine Ausgabe berechnen. Die Softmax-Aktivierungsfunktion wird verwendet, um die Ausgabe als Wahrscheinlichkeitsverteilung für jede Klasse zu normalisieren. Das Netz gibt dann die Klasse mit der höchsten Wahrscheinlichkeit als Vorhersage für das Eingabebild aus Basha et al. (2020).

2.1.2 Training von CNNs

Das Training von Convolutional Neural Networks (CNNs) erfolgt üblicherweise durch den Einsatz von großen Mengen annotierter Bilder, um das Modell darauf zu trainieren, bestimmte Merkmale zu erkennen und Entscheidungen zu treffen. Der Trainingsprozess besteht aus mehreren Schritten, die iterativ durchgeführt werden. Zunächst wird das Netzwerk mit einer zufälligen Initialisierung der Gewichte (z.B. Xavier Initialisierung) gestartet. Dann werden die Trainingsbilder in das Netzwerk eingespeist und die Ausgabe des Netzes mit den tatsächlichen Labels der Bilder verglichen. Der Fehler zwischen der Netzausgabe und den tatsächlichen Labels wird berechnet und die Gewichte des Netzes werden so angepasst, dass der Fehler minimiert wird. Dieser Prozess kann mithilfe von backpropagation, einer bekannten Optimierungsfunktion, realisiert werden Narkhede et al. (2022).
Während des Trainingsprozesses werden die Gewichte des Netzes schrittweise optimiert, um die Leistung des Modells zu verbessern. Eine häufig verwendete Optimierungsmethode ist der Gradientenabstiegsalgorithmus, der es dem Modell ermöglicht, sich Schritt für Schritt der optimalen Lösung anzunähern.Harshit et al. (2018) Die Schrittweite des Algorithmus wird durch den Lernrate-Hyperparameter gesteuert, der typischerweise im Voraus festgelegt wird und während des Trainings ebenfalls optimiert werden kann.
Das Training von CNNs kann sehr rechenintensiv sein, insbesondere bei Verwendung großer Datenmengen und komplexer Netzwerkarchitekturen. Zur Reduzierung der Trainingszeit können Grafikkarten (GPUs) eingesetzt werden, um die Berechnungen zu beschleunigen. Darüber hinaus können vortrainierte Modelle verwendet werden, um das Training zu optimieren und die Genauigkeit des Modells zu verbessern. Transfer Learning wird im Kapitel 2.4 erläutert.

Optimierungsfunktionen

Optimierungsfunktionen dienen dazu das Netz beim Training zu unterstützen und die optimalen Gewichte zu finden. Hierzu gibt es den Backpropagationalgorithmus, der die Optimierung von künstlichen neuronalen Netzen (KNN) unterstützt. Es ist der am häufigsten verwendete Algorithmus für das Training von Deep Learning-Modellen. Der Algorithmus nutzt eine Gradientenabstiegsmethode, wie z.B. adam Kingma and Ba (2014), um die Gewichte des KNNs schrittweise zu aktualisieren und dadurch die Vorhersagegenauigkeit des Netzes zu verbessern.

Die Backpropagation-Technik beinhaltet die Vorwärts- und Rückwärtsdurchläufe des Netzes. Während des Vorwärtsdurchlaufs wird die Eingabe durch das Netz propagiert und eine Ausgabe erzeugt. Während des Rückwärtsdurchlaufs wird der Fehler zwischen der Ausgabe und dem erwarteten Ergebnis berechnet und zurück durch das Netz propagiert, um die Gewichte zu aktualisieren.

In der Praxis hat sich der Optimierungsalgorithmus adam bei neuronalen Netzen bewährt. Adam (Adaptive Moment Estimation) ist ein Algorithmus für Gradienten-basiertes Lernen

beim maschinellen Lernen, der eine adaptive Schrittweite für jedes Parameter-Update berechnet und somit schneller konvergiert als herkömmliche stochastische Gradientenabstiegsverfahren (SGD).

Adam verwendet zwei Momente: den exponentiell abfallenden Durchschnitt der bisherigen Gradienten (1. Moment) und den exponentiell abfallenden Durchschnitt der bisherigen Gradientenquadratwerte (2. Moment). Der Algorithmus kombiniert diese beiden Momente, um eine adaptive Lernrate für jedes Gewicht zu berechnen. Wie beim SGD werden die Gradienten berechnet und die Gewichte aktualisiert, wobei Adam auch eine Schrittweitenkorrektur vornimmt, um eine Verzerrung der Schrittweite während der Anfangsphase des Lernens zu vermeiden.

Adam berechnet für jedes Gewicht im Modell einen adaptiven Schrittweitenparameter, der aus einem gleitenden Durchschnitt der vorherigen Gradienten und des vorherigen Quadrats der Gradienten berechnet wird. Diese Parameter werden als Schätzungen für den ersten und zweiten Moment des Gradientenvektors interpretiert, wobei der erste Moment der Durchschnitt der Gradienten und der zweite Moment der Durchschnitt der Quadrate der Gradienten ist. Adam berechnet eine effektive Lernrate für jedes Gewicht durch Division des ersten Moments durch die Wurzel des zweiten Moments und skaliert dies mit einem Schrittweitenkorrekturterm, um eine Verzerrung der Schrittweite während der Anfangsphase des Lernens zu vermeiden.

Adam bietet mehrere Vorteile gegenüber anderen Optimierungsalgorithmen, wie z.B. eine effektive Verwaltung der Schrittweite für jedes Gewicht, die Anpassung an unterschiedliche Skalen der Gradienten und die Möglichkeit, Hyperparameter automatisch anzupassen. Adam hat sich als effektiver Optimierungsalgorithmus für eine Vielzahl von Anwendungen im maschinellen Lernen erwiesen, wie z.B. für die Bild- und Spracherkennung.

Aktivierungsfunktionen

Es gibt unterschiedliche Aktivierungsfunktionen für unterschiedliche Anwendungsfälle. Aktivierungsfunktionen sind in allen Klassifikations-Layern eines Netzes zu finden. Besondere Beachtung gilt hierbei einer spezifischen Funktion, die für die Klassifikation auf den Bildern verantwortlich ist.
Die Aktivierungsfunktion **Softmax** ist eine der häufig verwendeten Funktionen in der künstlichen Intelligenz, insbesondere in der Deep-Learning-Forschung. Sie wird verwendet, um die Ausgabe einer Schicht von Neuronen in einen Vektor von Wahrscheinlichkeiten zu transformieren. Die Softmax-Funktion wird in der Regel in neuronalen Netzen verwendet, um die Ausgabe von Neuronen zu normalisieren. Sie ist besonders nützlich für Mehrklassenprobleme, bei denen eine Eingabe einer von mehreren Klassen zugeordnet werden muss. Sie ist auch eine Verallgemeinerung der logistischen Funktion (auch bekannt als Sigmoid-Funktion), die eine binäre Klassifikation ermöglicht. Sie kann auf eine beliebige Anzahl von Klassen erweitert werden. Vereinfacht gesagt, wird die Softmax-Funktion im Klassifikationsteil des Netzen, also unmittelbar nach den Pooling-Layern und vor dem Output-Layer, verwendet, um die gewünschten Klassen auf dem Bild zu identifizieren und Wahrscheinlichkeiten auszugeben. Die Anwendung von Softmax eignet sich für die Klassifizierung von Bildern, bei der jedes Bild einer von mehreren Kategorien zugeordnet werden muss, zum Beispiel Hunde, Katzen oder Vögel.

Die Softmax-Funktion hat mehrere wichtige Eigenschaften. Eine davon ist, dass die Ausgabe von Softmax immer eine Wahrscheinlichkeitsverteilung ist. Das bedeutet, dass die Summe der Ausgabevektorelemente 1 ist und jeder Wert zwischen 0 und 1 liegt.

Regularization und Data Augmentation

Um das Problem des Overfittings in der Bildklassifikation zu vermeiden, gibt es verschiedene Techniken, die angewendet werden können, um die Generalisierungsfähigkeit des Modells zu verbessern. Zwei häufig verwendete Techniken sind Regularisierung und Data Augmentation.

Regularisierung ist ein Ansatz zur Verbesserung der Generalisierungsfähigkeit eines Modells, bei dem zusätzliche Bedingungen in die Trainingsverlustfunktion integriert werden. Diese Bedingungen helfen dabei, die Anzahl der zu optimierenden Gewichte zu reduzieren und somit das Overfitting-Risiko zu minimieren. Eine häufig verwendete Regularisierungstechnik in CNNs ist das sogenannte Dropout-Verfahren. Dabei werden während des Trainingsprozesses zufällig ausgewählte Neuronen in einer Schicht temporär deaktiviert. Dadurch wird das Netz dazu gezwungen, Redundanzen in der Repräsentation der Daten zu entwickeln und das Overfitting-Risiko zu verringern. Srivastava et al. (2014).

L2-Regularisierung, auch bekannt als Ridge-Regularisierung oder Tikhonov-Regularisierung, ist eine weitere häufig verwendete Regularisierungstechnik in CNNs neben Dropout. Sie fügt der Verlustfunktion einen Strafterm hinzu, der proportional zur Summe der quadrierten Gewichte im Modell ist. Durch die Integration von L2-Regularisierung in die Verlustfunktion werden größere Gewichte im Modell bestraft, was dazu beiträgt, die Anzahl der Gewichte zu reduzieren, die optimiert werden müssen, und die Generalisierungsfähigkeit des Modells zu verbessern. Im Gegensatz zu Dropout reduziert L2-Regularisierung die Gewichtswerte selbst, anstatt Neuronen auszuschalten. Phaisangittisagul (2016).

Data Augmentation Data Augmentation ist eine Technik, die in der Bildklassifikation häufig eingesetzt wird, um synthetische Trainingsdaten zu generieren. Dabei werden Transformationen auf die vorhandenen Trainingsbilder angewendet, um das Netz auf eine größere Vielfalt von Bildern vorzubereiten. Durch Zufallsrotationen, Skalierungen und Spiegelungen der Trainingsbilder wird das Netz dazu gebracht, mehrere Varianten von Bildern zu erkennen und zu klassifizieren. Data Augmentation kann auch dazu beitragen, die Größe des Trainingsdatensatzes zu erhöhen, um das Risiko von Overfitting zu reduzieren, insbesondere wenn der Datensatz klein ist. Taylor and Nitschke (2018)

In der Praxis werden Regularisierung und Data Augmentation oft gemeinsam verwendet, um die Leistung von CNNs in der Bildklassifikation zu verbessern. Es ist jedoch wichtig zu beachten, dass eine zu starke Anwendung von Regularisierung und Data Augmentation das Modell daran hindern kann, komplexe Muster in den Daten zu erkennen, was zu einer verringerten Genauigkeit führen kann. Daher ist es wichtig, die Regularisierungs- und Augmentierungstechniken sorgfältig zu wählen und zu konfigurieren, um ein optimales Gleichgewicht zwischen Leistung und Generalisierungsfähigkeit zu erreichen.

2.1.3 Lernmethoden

Supervised learning

Im Bereich der visuellen Informationsverarbeitung wird oft die Methode des Supervised learning eingesetzt. Hierbei werden Trainingsdaten visuell annotiert und bestimmten Kategorien (z.B. Autos, Lebensmittel, Gebäude) zugeordnet. Mit diesen Daten wird ein Algorithmus trainiert, der anhand von visuellen Merkmalen eine automatische Pixel-basierte Klassifizierung des gesamten Bildes ermöglicht.

Um den Algorithmus zu optimieren, werden die Trainingsdaten üblicherweise in Trainings- und Validierungssätze aufgeteilt. Während der Trainingsdatensatz zur Trainingsphase dient, wird der Validierungssatz zur Bewertung und Optimierung der Leistung genutzt.

Supervised learning wird in vielen Anwendungsbereichen, wie der Fernerkundung, medizinischen Bildverarbeitung oder der Automobilindustrie eingesetzt, um visuelle Informationen

effektiv zu verarbeiten und zu interpretieren.

Unsupervised learning

Unsupervised learning ist eine Methode zur automatischen Klassifizierung von Bildern, die ohne annotierte Trainingsdaten auskommt. Stattdessen werden Algorithmen eingesetzt, um spezifische Merkmale in einem Bild während der Bildverarbeitung zu erkennen. Diese Merkmale werden genutzt, um die Bildpixel in verschiedene Gruppen oder Klassen zu unterteilen. Dabei werden ähnliche Pixel in derselben Gruppe zusammengefasst.

Unsupervised learning findet Anwendung, um unbekannte oder komplexe Muster in Bildern zu erkennen und zu gruppieren. Dabei wird es oft in der Bildverarbeitung eingesetzt. Aber auch in der Überwachung von Umweltveränderungen oder der automatischen Identifizierung von Objekten und Regionen in Bildern wird die Methode genutzt.

2.2 Image Classification

Die automatische Klassifizierung von Bildern, auch als Image Classification bezeichnet, bezieht sich auf den Prozess, bei dem ein Computer die Kategorie oder Klasse eines Bildes bestimmt. Vor der Entwicklung von Deep Learning-Algorithmen war es schwierig, Bilder durch Algorithmen allein zu klassifizieren. Dank der Fortschritte im Deep Learning kann ein Computer heute mit einer Genauigkeit vergleichbar mit der eines Menschen bestimmen, ob sich auf einem Bild beispielsweise ein Auto oder ein Apfel befindet. Dies wird erreicht, indem Gruppen von Pixeln oder Vektoren innerhalb eines Bildes unter bestimmten Regeln kategorisiert werden. Die Regeln können auf der Grundlage eines oder mehrerer struktureller Merkmale aufgestellt werden.

Relation zwischen Image Classification und Object Detection

Image Classification und Object Detection sind eng miteinander verwandte Aufgaben der Computer Vision, bei denen es darum geht, Muster und Merkmale in Bildern automatisch zu erkennen und zu klassifizieren. Beide Aufgaben verwenden Deep-Learning-Modelle, insbesondere Convolutional Neural Networks (CNNs), als Kernkomponenten für die Merkmalsextraktion und Klassifizierung.

Sowohl bei der Object Detection als auch bei der Supervised Image Classification geht es darum, ein Modell zu trainieren, das in der Lage ist, komplexe Muster und Merkmale in Bildern zu erkennen und zu klassifizieren.

In einigen Fällen werden vortrainierte Image Classification-Modelle als Ausgangspunkt für die Entwicklung von Object Detection-Modellen verwendet.

Die Komplexität eines Object Detection-Modells ist höher als die eines einfachen Klassifikationsmodells, da es nicht nur Klassifikationsaufgaben bewältigen, sondern auch die Position eines Objekts im Bild bestimmen muss, bevor eine Klassifikation durchgeführt werden kann. Dadurch kann es vorkommen, dass bestimmte Objekte im Bild nicht erkannt werden, obwohl der Klassifikationsteil des Modells in der Lage wäre, diese zu identifizieren. Das liegt daran, dass das Modell möglicherweise nicht in der Lage war, die Position des Objekts korrekt zu bestimmen und daher keine Klassifikation durchführen konnte.

In der Praxis bezieht sich Object Detection auf eine Methode, bei der ein neuronales Netzwerk eingesetzt wird, um bestimmte Bereiche in einem Bild zu markieren und diese Teilbereiche anschließend an ein Klassifikationsnetz weiterzugeben. Im Rahmen der Image Classification ist jedem Datensatz ein eigenes Label zugeordnet, das die Klasse enthält. Ein ganzes Bild stellt somit den Wert x dar, während das dazugehörige Label y die Klasse

bezeichnet. Das Modell wird dazu genutzt, um festzustellen, was auf einem bestimmten Bild zu sehen ist. Angenommen, das Modell hat die Wahl zwischen den drei Klassen 'Hund', 'Katze' und 'Maus'. Sollte das Modell eine 'Katze' erkennen und somit falsch liegen, wird es anhand des Labels 'Hund' darüber informiert, dann passt es seine Gewichte entsprechend an. Bei mehreren Bildern werden mehrere x-Werte als Liste an das Modell übergeben, während die dazugehörigen y-Werte in einer anderen Liste gespeichert werden. Auf diese Weise kann das Modell jedem Bild eine entsprechende Klasse zuordnen.
Im Rahmen der Object Detection wird diese Methode jedoch erweitert. Neben der Klasse wird nun auch die Position wiedergegeben. Darüber hinaus können auf einem Bild mehrere Klassen gleichzeitig vorhanden sein, was eine Multi-Label-Klassifikation erforderlich macht. Nun wird nicht mehr eine Liste mit x-Werten, beispielsweise mit den Pfaden zum jeweiligen Bild, und eine Liste mit y-Werten mit den ausgeschriebenen Klassen an das Modell übergeben, stattdessen werden mehrdimensionale Listen für den y-Wert verwendet. Die Bilder werden erneut als Liste mit x-Werten übergeben. Jedes x hat ein y, welches wiederum eine Liste mit Informationen enthält. Jeder einzelne y-Wert enthält die Position der Bounding Box eines Objekts sowie das dazugehörige Label. Auf diese Weise können für jedes Bild die entsprechenden Bounding Boxen, bestehend aus Koordinaten und den zugehörigen Klassen, zu dieser Position im Bild ermittelt werden. Ein Object Detection-Modell hat die Aufgabe, verschiedene Objekte auf einem Bild zu finden, diesen eine Bounding Box zuzuordnen und anschließend anhand der zuvor übergebenen y-Werte zu prüfen, ob die Bounding Box auf einem Objekt liegt. Nachdem die Bounding Box überprüft wurde, wird dieses Teilbild an ein Klassifikationsnetz weitergegeben, um zu prüfen, welche Klasse sich dahinter verbirgt. Hier wird wiederum auf das ursprüngliche y zugegriffen, um zu prüfen, ob das Teilbild die gewünschten Klassen entspricht.

2.3 Object Detection

Die Erkennung und Klassifizierung von mehreren Objekten in einem Bild wird als Object Detection bezeichnet. Ziel ist es, alle Objekte einer bestimmten Klasse, wie beispielsweise Personen, Autos oder Gesichter, zu identifizieren, unabhängig von ihrer Größe und Position im Bild. Die Erkennung liefert Informationen über das Objekt, wie beispielsweise Position, Größe, Begrenzungsrahmen oder Segmentierungsmaske. Mithilfe von Trainingsdaten kann ein Objekterkennungs Modell erstellt werden, das die Merkmale und Charakteristika einer bestimmten Klasse lernt. Dabei wird ein Klassifizierungsmodell entwickelt, das in der Lage ist, Objekte in Bildern korrekt zu erkennen und zu klassifizieren. Die Qualität des Modells hängt von der Menge und Qualität der verfügbaren Trainingsdaten ab Amit et al. (2020). Objekterkennung wird in verschiedenen Anwendungsgebieten wie der Medizin Li et al. (2019), dem autonomen Fahren Wang et al. (2019), der Überwachung und der Lösung visueller Probleme Antol et al. (2015) eingesetzt. Bei der Objekterkennung gibt es zwei Arten von Detektoren: One-Stage-Detektoren und Two-Stage-Detektoren.

2.3.1 One-stage Detector

One-stage detector schieben eine komplexe Anordnung möglicher Bounding Boxes, Anker genannt, über das Bild und klassifizieren sie direkt, ohne den Inhalt der Box anzugeben. Der One-Stage Detector geht davon aus, dass in einem bestimmten Bereich des Bildes möglicherweise ein Objekt vorhanden ist, aber er weiß nicht genau, welches Objekt. Daher werden die Anchors über das gesamte Bild geschoben und für jeden Anchor wird ein Vorhersagewert für die Wahrscheinlichkeit berechnet, ob und wo sich in dem Bereich ein

bestimmtes Objekt befindet. Lin et al. (2017b), Redmon et al. (2016)

2.3.2 Two-stage Detector

Zunächst werden im Object detection Prozess 'region proposals' generiert, also Vorschläge für begrenzte Regionen auf dem Bild, in denen sich mit höherer Wahrscheinlichkeit Objekte befinden könnten. Es gibt verschiedene Techniken zur Generierung dieser Vorschläge, die auf bestehenden Methoden wie Region Based Convolutional Neural Networks (R-CNN) Girshick et al. (2014) oder Feature Pyramid Networks (FPN) Lin et al. (2017a) basieren können.

Im zweiten Schritt wird die eigentliche Objekterkennung auf den generierten Region Proposals durchgeführt. Ein CNN wird verwendet, um die Klassifizierung und Lokalisierung des Objekts in jedem Region Proposal durchzuführen. Das CNN wird einzeln auf jedem Region Proposal angewendet. Ren et al. (2015), Girshick (2015).

Gemäß Ren et al. (2015) bietet die Two-Stage Detection den Vorteil, dass nur eine begrenzte Anzahl von Regionen genauer untersucht werden muss, wodurch der Prozess beschleunigt wird. Darüber hinaus können diese Region Proposals gezielt ausgewählt werden, um die Erkennungsgenauigkeit zu maximieren.

2.3.3 CenterNet

CenterNet ist ein Ansatz zur Objekterkennung, der darauf basiert, dass Objekte als einzelne Punkte - nämlich die Mittelpunkte ihrer Bounding Boxes - modelliert werden. Im Gegensatz zu herkömmlichen Objekterkennungsprogrammen, die eine lange Liste potenzieller Objektpositionen durchgehen und jedes einzelne klassifizieren, verwendet CenterNet Keypoint-Schätzung, um den Mittelpunkt des Objekts zu finden und auf andere Eigenschaften des Objekts zu schließen, wie z.B. Größe, 3D-Position, Ausrichtung und Pose.
Dank der durchgängigen Differenzierbarkeit ist CenterNet schneller, einfacher und genauer als andere Bounding-Box-basierte Detektoren. CenterNet erreicht tatsächlich das beste Verhältnis zwischen Geschwindigkeit und Genauigkeit im MS COCO-Datensatz mit 28,1% AP bei 142 FPS, 37,4% AP bei 52 FPS und 45,1% AP mit Multiskalentests bei 1,4 FPS. Die Methode von CenterNet ist konkurrenzfähig mit ausgefeilten mehrstufigen Methoden und läuft in Echtzeit, was sie zu einer vielversprechenden Alternative für die schnelle und effiziente Objekterkennung macht. Zhou et al. (2019)

Ankerbasierte und Heatmap-basierte Detektoren

Aus der Perspektive der Objektbegrenzungsbox-Darstellung können die Detektoren in zwei Gruppen unterteilt werden: Ankerbasierte und Heatmap-basierte.
Ankerbasierte Detektoren wie R-CNN Girshick (2015) und andere, haben vordefinierte Bounding Boxes an jeder räumlichen Ausgangsposition, die als Ankerboxen bezeichnet werden. Diese Ankerboxen haben verschiedene Größen und Seitenverhältnisse, um eine Vielzahl von Objektformen abzudecken. Durch die Anpassung der Ankerboxen an die tatsächliche Position und Größe des Objekts wird die Position und Größe der Begrenzungsboxen geschätzt.
Heatmap-basierte Detektoren wie CornerNet Law and Deng (2018) und CenterNet Zhou et al. (2019) erkennen Schlüsselpunkte der Bounding Box eines Objekts, wie die Eckpunkte

oder das Zentrum. Das Netzwerk gibt Heatmaps der Keypoints und verschiedene Regressionswerte für den Offset oder die Rohgröße des Begrenzungsrahmens in Abhängigkeit von verschiedenen Strukturen aus. Um möglichst viele Objektformen abzudecken, werden bei ankerbasierten Methoden unterschiedliche Größen und Höhen/Breiten-Verhältnisse für Ankerboxen vorangestellt. Dies erhöht natürlich den Umfang der Ausgabe, was sich als rechenintensiv erweist. Somit bleibt der Kompromiss zwischen Robustheit und Geschwindigkeit schwer zu lösen. Die Heatmap-basierten Detektoren vermeiden dieses Dilemma jedoch, da alle Regressionswerte rohe Pixelkoordinatenwerte abbilden und keine Voreinstellungen erforderlich sind.

Von keypoint-Schätzung zur Boundingbox

Das heatmap-basierte OD-Modell CenterNet stellt eine innovative Methode dar, um Objekte als einen Punkt in ihrem Zentrum darzustellen. Dieses Modell erzeugt eine Heatmap, indem das Eingangsbild durch ein CNN verarbeitet wird. Die Spitzenwerte in dieser Heatmap entsprechen den Objektzentren, während die Bildmerkmale an jedem Spitzenwert der Heatmap die Höhe und das Gewicht der Boundingbox eines Objekts vorhersagen.
Während der Inferenzzeit (Vorhersagezeit) werden zunächst die Spitzenwerte der Heatmap für jedes Objekt extrahiert und als Keypoints notiert. Hierbei werden nur die Ergebnisse als Keypoints behalten, die größer oder gleich ihren acht Nachbarn sind. Die Top hundert Spitzenwert-Ergebnisse werden anschließend ausgewählt und jeder erstellte Keypoint kann in (x,y)-Koordinaten repräsentiert werden. Das Keypoint-Set wird dann als Indikator für die Erkennungswahrscheinlichkeit verwendet und an dieser Stelle wird eine Boundingbox erstellt. Zhou et al. (2019)

Abbildung 2.3: CenterNet Output for OD, Zhou et al. (2019)

2.4 Transfer Learning

Transfer Learning ist eine Technik des maschinellen Lernens, bei der ein bereits trainiertes Modell als Ausgangspunkt für das Training eines neuen Modells verwendet wird. Statt ein Modell von Grund auf neu zu trainieren, nutzt man das bereits trainierte Modell als Basis, um ein neues Modell zu erstellen, das auf ähnlichen oder sogar verschiedenen Aufgaben angewendet werden kann.

Transfer Learning ermöglicht es, auf einfache Weise hochpräzise Modelle für neue Aufgaben zu erstellen, ohne dass man über ausreichende Datenmengen und Rechenleistung verfügen muss, um das Modell von Grund auf neu zu trainieren. Das bereits trainierte Modell, das als Ausgangspunkt verwendet wurde, verfügt über ein breites Spektrum an Fähigkeiten und Wissen, das auf das neue Modell übertragen werden kann, um es schneller und effektiver zu trainieren.

Transfer Learning wird in vielen Anwendungen des maschinellen Lernens eingesetzt, darunter Bilderkennung, Spracherkennung und natürliche Sprachverarbeitung. Es hat sich als äußerst nützlich erwiesen, um Modelle zu erstellen, die in der Lage sind, komplexe Aufgaben zu bewältigen und dabei eine hohe Genauigkeit und Effizienz zu erreichen.

2.4.1 Vergleich durch ein Beispiel aus der realen Welt

Dank Transfer Learning ist es möglich mit einer geringeren Datenmenge ein relativ gutes Ergebnis bei einem ähnlichen Problem zu erreichen. Die Alternative ist ein eigenes Netz zu erstellen und von Anfang an zu trainieren. Während die Gewichte in den Neuronen beim Transfer Learning bereits eingestellt sind und als Basis dienen, sind diese bei einem selbst erstellten Netz nicht eingestellt. Dies lässt sich mit dem menschlichen Gehirn vergleichen. Ein Kleinkind kann in der Regel noch nicht sonderlich gut sprechen, während eine erwachsene Person dies durch kontinuierliches Trainieren erlernt hat, also durch das Hören, Wiedergeben, Anwenden und Korrigieren. Die Menge und Qualität des Gelernten entscheiden darüber, wie gut das Kleinkind in Zukunft sprechen wird. Sprechen die Eltern viel mit dem Kind, wird es später beim Satzbau korrigiert, wird Zuhause ein großer Wortschatz verwendet, dann ist die Wahrscheinlichkeit groß, dass es später besser sprechen wird als andere Kinder, bei denen es nicht der Fall ist Jampert et al. (2009). Möchte man einem Kind die Aufgabe geben, einen Vortrag in höherem Deutsch zu halten, wird es dem Kind wahrscheinlich schwer fallen. Eine erwachsene Person hat die Grundlagen bereits gelernt und muss lediglich weitertrainieren, um die Sprache zu verbessern. Genauso verhält sich ein untrainiertes eigenes Netz im Gegensatz zu einem vortrainierten. Für das untrainierte Netz benötigt man somit mehr Daten zum Trainieren als bei einem trainierten Netz.

2.4.2 Verstehen und nutzen eines vortrainierten Modells

Ein pre-trained model kann ohne weitere Änderungen am Netz oder Code verwendet werden. Hierfür muss nur der Output Layer verstanden werden, in welcher Form die predictions geliefert werden.Er ist bei Bedarf anpassen. Jedes Netz hat eine sog. labelmap, in der die Klassen, die das Modell kennt, enthalten sind. Die labelmap ist eine einfache und menschenlesliche Textdatei. Als Beispiel kann das Netz CenterNet genommen werden. Dieses wurde mit dem COCO-2017 Datensatz trainiert und erkennt unter anderem Personen auf Bildern. Es ist somit möglich Bilder nach Personen zu durchsuchen, ohne größeren Aufwand. Möchte man das Netz verbessern oder auf andere Anwendungen umtrainieren, können die Methoden Feature Extraction oder Fine Tuning angewendet werden. Diese Methoden werden im Folgenden erläutert.

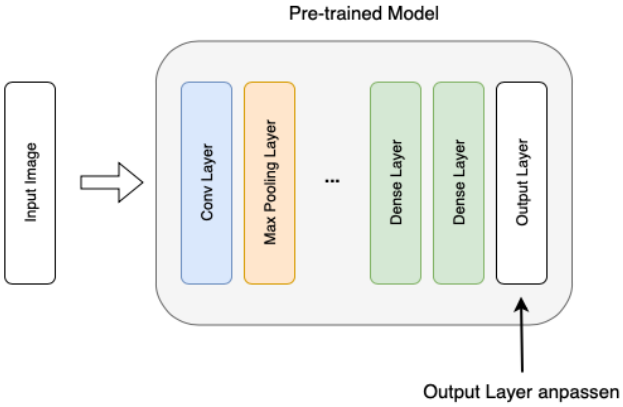

Abbildung 2.4: Pretrained Model 1

2.4.3 Feature Extraction

Im Kontext von Fine Tuning bezieht sich Feature Extraction auf den Prozess, ein vortrainierts Modell zu nehmen und nur einen Teil davon zu trainieren, um Features (Merkmale) aus den Daten zu extrahieren. Die extrahierten Features werden dann als Eingabe an weitere Layer, die für die Klassifikation der Objekte zuständig sind, übergeben, um das Netz auf eine spezifische Aufgabe anzupassen. Dazu wird der Teil des Netzes, der für die feature extraction zuständig ist, eingefroren. Somit wird sichergestellt, dass nur der Klassifikationsteil des Netzes trainiert wird, wie im Bild 2.5 dargestellt.

Typischerweise verwendet man vortrainierte Netze wie Convolutional Neural Networks (CNNs) oder Recurrent Neural Networks (RNNs), um Features aus Bildern, Texten oder anderen Daten zu extrahieren. Diese vortrainierten Netze wurden auf großen und vielfältigen Datensätzen trainiert und können daher bereits eine bestimmte Menge von Merkmalen und Mustern erkennen. Zum Beispiel kann ein vortrainiertes CNN, das auf die Bildklassifizierung trainiert wurde, als Feature-Extraktor verwendet werden, um Merkmale aus einem Datensatz von medizinischen Bildern zu extrahieren. Diese Merkmale können dann als Input für ein weiteres Modell dienen, das speziell auf die Erkennung von Krankheiten oder Anomalien in medizinischen Bildern trainiert ist. Durch die Verwendung von Feature Extraction können Entwickler schnell und effizient vortrainierte Modelle nutzen und anpassen, um spezifische Aufgaben zu erfüllen, ohne das gesamte Modell neu trainieren zu müssen.

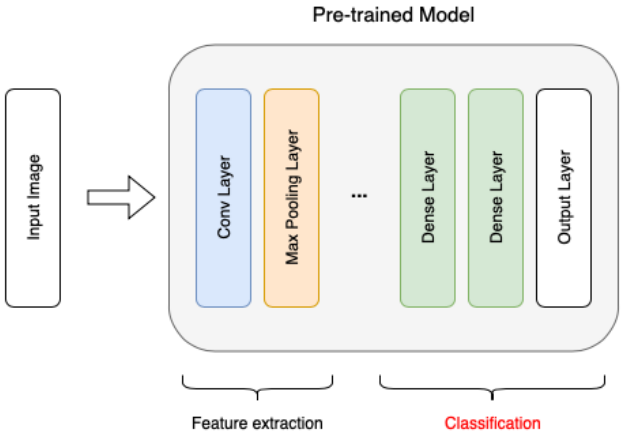

Abbildung 2.5: Pretrained Model 2

Zusätzlich kann man durch die Verwendung von Feature Extraction Zeit und Rechenressourcen sparen, da man nicht das gesamte vortrainierte Modell neu trainieren, sondern nur den zusätzlichen Klassifikationsteil anpassen muss, um die spezifische Aufgabe zu lösen. Feature Extraction kann auch dazu beitragen, Overfitting zu vermeiden, da man das vortrainierte Modell bereits auf einer großen Anzahl von Daten trainiert und somit bereits eine gute allgemeine Fähigkeit hat, Merkmale zu erkennen. Im Bild 2.6 wird ein Training mit zusätzlicher Validation dargestellt. In diesem Beispiel steigt die Accuracy schon nach wenigen Epochen auf über 90%.

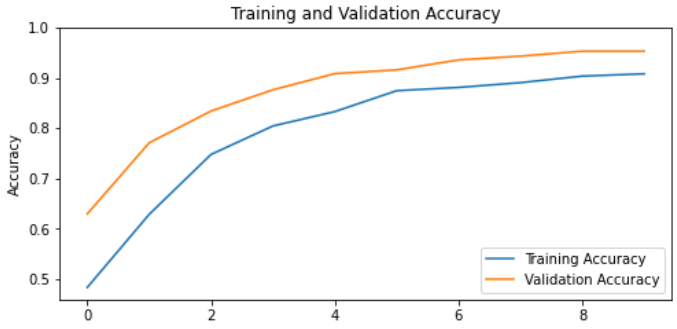

Abbildung 2.6: Accuracy

2.4.4 Fine Tuning

Beim Fine Tuning werden die vortrainierten Modelle auf eine neue spezifische Aufgabenstellungen angepasst, indem die Gewichte des Modells auf Daten des neuen Anwendungsbereichs trainiert werden. Fine Tuning erfordert in der Regel mehr Zeit und Rechenressourcen als Feature Extraction, kann aber auch zu besseren Ergebnissen führen, da das Modell speziell für die neue Aufgabe optimiert ist.

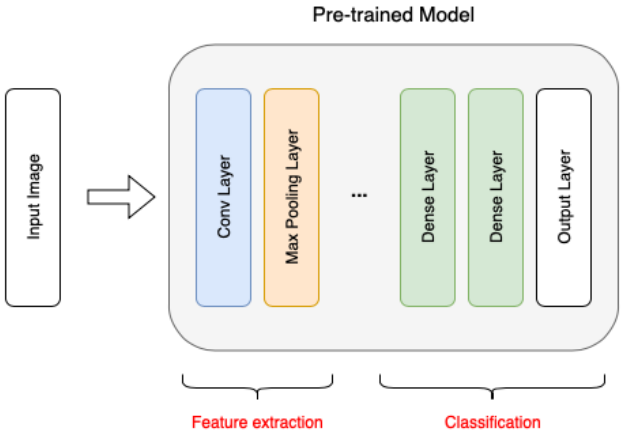

Abbildung 2.7: Pretrained Model 3

Allerdings sollte dies erst versucht werden, nachdem der Klassifikationsteil trainiert wurde, wobei im vortrainierten Netz der Feature Extraction Teil eingefroren wurde. Wenn ein zufällig initialisierter Klassifikator zu einem bereits trainierten Modell hinzufügt wurde und versucht wird, alle Schichten gemeinsam zu trainieren, wird die Größe der Gradientenaktualisierungen zu groß sein (aufgrund der zufälligen Gewichte des Klassifikators) und das bereits trainierte Modell wird vergessen, was es gelernt hat. Folgendes Bild zeigt zur Veranschaulichung die Genaugkeit nach Anwendung von Fine Tuning. Diesmal wurde festgelegt die letzten vierundfünzig von huntervierundfünfzig Layer zu trainieren, um die Gradientenaktualisierung nicht zu groß werden zu lassen.

Abbildung 2.8: Accuracy 2

2.4.5 Gradientenaktualisierung

Gradientenaktualisierung ist ein zentraler Schritt in vielen Optimierungsalgorithmen, die bei der Anpassung von Modellen im maschinellen Lernen verwendet werden. Dabei werden die Gewichte eines Modells basierend auf der Größe und Richtung des Gradienten, der aus der Verlustfunktion des Modells berechnet wird, angepasst.

Der Gradient einer Funktion ist eine Vektormenge, die die Richtung und Steigung der Funktion an einem bestimmten Punkt angibt. Im Kontext des maschinellen Lernens ist der Gradient der Verlustfunktion des Modells ein Vektor, der die Richtung und Steigung der Verlustfunktion an einem bestimmten Punkt im Gewichtsraum angibt.

Der Gradientenaktualisierungsschritt beinhaltet die Berechnung des Gradienten der Verlustfunktion des Modells in Bezug auf die Gewichte und die Aktualisierung der Gewichte des Modells in Richtung des negativen Gradienten. Das Ziel besteht darin, den Wert der Verlustfunktion zu minimieren, indem man iterativ die Gewichte des Modells anpasst.

Es gibt verschiedene Optimierungsalgorithmen, die auf der Gradientenaktualisierung basieren, wie z.B. Stochastic gradient descent (SGD), der Adaptive Moment Estimation (Adam) oder der RMSprop Kurbiel and Khaleghian (2017). Jeder dieser Algorithmen hat seine eigenen spezifischen Anpassungsregeln und Hyperparameter, die darauf abzielen, eine schnelle und effektive Konvergenz des Modells zu erreichen. Mehr Informationen zum Optimierungsalgorithmus Adam im Kapitel 2.1.2.

3 Praktische Anwendung Object Detection im Kontext einer ausgewählten Bild-Datenmenge des Badischen Landesmuseum

3.1 Daten

Die zugrundeliegenden Daten stammen aus der Bildersammlung 'Staufen', welche vom Museum zur Verfügung gestellt wurde. Die Daten wurden unter sorgfältigen Aufbewahrungsbedingungen verwahrt, um deren Integrität und Verfügbarkeit zu gewährleisten. Aus Datenschutzgründen dürfen die Daten nicht ins Internet gelangen. Aus diesem Grund wurde nur die Hardware genutzt, die zur Verfügung stand. Es wurden keine Tools verwendet, die in der Cloud ausgeführt werden.

3.1.1 Erkunden der Daten

Ein wichtiger Bestandteil dieser Arbeit ist die Verwendung eines einzigartigen Datensatzes.

Umfang und Größe des Datensatzes

Die Gesamtzahl der gescannten Bilder betrug 9121. Beim Trainieren eines Modells ist es in der Regel empfehlenswert, mit einer großen Datenmenge zu trainieren, um die Genauigkeit zu verbessern. Die zur Verfügung gestellten Bilder enthielten zu etwa 60% Bilder von Personen, wobei einige Bilder mehrere Personen unterschiedlichen Alters enthielten. Unter den Bildern, bei denen die Identifizierung wichtig war, waren auch Kinder. Die Gesamtzahl der Kinder in allen Bildern betrug 2.645. Viele dieser wichtigen Bilder von Kindern wurden von den vortrainierten Modellen in der Klasse 'person' erkannt. Allerdings wurden auch andere Klassen wie 'man' und 'woman' als 'person' erkannt. Diese beiden Klassen waren in dem Datensatz in folgender Anzahl vorhanden: 'man': 2808 Bilder, 'woman': 1883 Bilder.

Datensatz-Charakteristika

Farbtiefe: So hatten viele Bilder nur einen Farbkanal (Schwarz-Weiß-Fotos), während einige Bilder drei Farbkanäle hatten.

Dateigröße: 3262 Bilder befinden sich im Bereich 100 KB bis 1 MB während 2222 Bilder im Bereich von 10 KB - 100 KB liegen.

Dateiformat: Zur verfügung wurden die Bilder in JPEG und TIFF gestellt. Es wurde entschieden entschieden, die JPEG Bilder zum Trainieren des Modells zu verwenden, da alle Object Detection (OD) Modelle dieses Format nativ unterstützten. Die zusätzlichen Metainformationen von TIFF wären nicht hilfreich gewesen.

3.1.2 Daten Vorverarbeitung

Annotieren der Daten

Alle Bilder wurden zu Beginn des Projekts mithilfe eines vortrainierten OD Modells mit der Klasse 'person' vor-annotiert (vor-gelabelt). Das bedeutet, dass das Modell automatisch alle Personen in den Bildern erkannt und markiert hat. Anschließend wurden die Annotationen der Bilder in Label-Studio in eine der drei Klassen 'man', 'woman' oder 'child' umgeändert. Dieser manuelle Annotierungsprozess war zeitaufwendig und erforderte viel Aufmerksamkeit und Genauigkeit, da es wichtig war, die Personen auf den Bildern korrekt zu klassifizieren, um ein genaues Modell zu trainieren.

Datenqualität erhöhen

Die Effektivität von KI-Modellen hängt maßgeblich von der Qualität der zugrunde liegenden Daten ab. Daten von schlechter Qualität können zu ungenauen Modellen führen, die in realen Anwendungen nicht funktionieren. Im Gegensatz dazu führt eine hohe Datenqualität zu genaueren und leistungsfähigeren Modellen. Aus diesem Grund ist es entscheidend, die Qualität der Daten zu sichern und beispielsweise Ausreißer zu entfernen.
Um die Qualität von Daten zu gewährleisten, werden häufig verschiedene Maßnahmen ergriffen. Eine davon ist die Überprüfung von Annotationen in Datensätzen mithilfe von speziellen Skripten. Diese Skripte können mögliche Fehler in den Daten aufdecken und beheben. Ein Beispiel hierfür ist die Überprüfung von Bounding Box-Annotationen in Bildern, um sicherzustellen, dass sie sich innerhalb des jeweiligen Bildes befinden.
In der vorliegenden Arbeit wurde die Qualität der Daten durch die Überprüfung von Annotationen in einem umfassenden JSON-Format gewährleistet. Ein Python-Skript wurde entwickelt und angewendet, um mögliche Fehler in den Daten aufzudecken. Einige der aufgedeckten Fehler betrafen Annotationen, deren Bounding Boxen außerhalb des Bildes lagen.

3.2 Hard- und Softwareumgebung

3.2.1 Hardware

Wieso macht es keinen Sinn Netze rein mit der CPU zu trainieren? Weil die CPU allein nicht genügend Rechenleistung und Speicherbandbreite hat, um komplexe Modelle zu trainieren, die große Datensätze verarbeiten müssen. KI-Modelle bestehen in der Regel aus einer großen Anzahl von Neuronen und Schichten, die miteinander verbunden sind und während des Trainings sehr große Mengen an Daten verarbeiten müssen. Dieser Prozess erfordert eine erhebliche Rechenleistung und Speicherbandbreite, um die Berechnungen schnell und effizient durchzuführen. Im Gegensatz dazu bietet eine Grafikprozessoreinheit (GPU) eine viel höhere Rechenleistung und Speicherbandbreite, die für das Training von KI-Modellen benötigt wird. GPUs sind speziell für parallele Berechnungen optimiert und können mehrere Aufgaben gleichzeitig ausführen, was sie viel schneller macht als CPUs. Daher ist es sinnvoll, eine GPU für das Training von KI-Modellen zu verwenden, da dies die Geschwindigkeit des Trainingsprozesses erheblich verbessert und die Trainingszeit für komplexe Modelle reduziert. Ebenfalls wichtig ist der Speicher der GPU. Je mehr sie hat, desto mehr kann sie gleichzeitig in den RAM laden und schnell drauf zugreifen. Besonders bei komplexen Modellen, wie zum Beispiel Object Detection, mit vielen Parametern ist es zu empfehlen viel Speicher zu besitzen. Bei zu wenig Speicher muss ausgelagert werden, was die Performance beträchtlich schmälert.

Optimale Hardware

Die beste Hardware zum Trainieren von KI-Modellen hängt von verschiedenen Faktoren ab, wie der Größe des Datensatzes, der Komplexität des Modells und den verfügbaren Ressourcen. Für kleinere Datensätze und weniger komplexe Modelle kann eine Standard-CPU ausreichend sein. Wenn jedoch größere Datensätze oder komplexere Modelle verwendet werden, kann eine TPU verwendet werden, die speziell für die Verarbeitung von Tensoroperationen in neuronalen Netzen optimiert ist. Google Cloud bietet zum Beispiel Cloud TPUs als Cloud-Service an. Somit kann sehr viel Zeit bei der Entwicklung des gewünschten Modells gespart werden.

Verwendete Hardware

Alle Netze die in dieser Arbeit genutzt wurden, sind mit einem Windows Computer mit installiertem WSL2, AMD Ryzen 5 3600 @ 3.6-4.0 GHz x 6 Prozessor, 32GB GDDR4 RAM und einer NVIDIA GeForce RTX 3060 mit 12 GB VRAM Grafikkarte, weitertrainiert worden.

Getestet wurde ebenfalls auf MacOS mit einem M1 Chip und einem Windows 10 Laptop mit Intel i5 Prozessor. Jedoch gab es dort Inkompatibilitäten.

3.2.2 Framework

Nach intensiver Recherche fiel die Entscheidung auf das Framework TensorFlow, da es am besten geeignet ist und als State-of-the-art in der Industrie gilt. TensorFlow verfügt von Haus aus über viele Funktionen, darunter solche zum Durchsuchen oder Verarbeiten von Daten sowie zum anschließenden Evaluieren der Ergebnisse. Die einfache Bedienbarkeit und Möglichkeit zum fine tunen vortrainierter Modelle waren die ausschlaggebenden Gründe für die Entscheidung zugunsten von TensorFlow.

3.2.3 Softwaretools

Um die Arbeit an diesem Projekt zu optimieren, wurde stets moderne Technologie eingesetzt - angefangen beim Betriebssystem, mit dem gearbeitet wurde, bis hin zur integrierten Entwicklungsumgebung (IDE), mit der Skripte entwickelt und ausgeführt wurden.

Betriebssystem

Als Betriebssystem wurde Windows 10 mit installiertem Linux-Subsystem Ubuntu verwendet, um bestimmte Befehle auszuführen, die in der Anleitung der TensorFlow Object Detection API enthalten waren.

Python Umgebung

Für die Python-Umgebung wurde Miniconda3 genutzt. Dadurch konnten Python-Pakete abgegrenzt werden, um Inkompatibilitäten entgegenzuwirken. Aufgrund der zahlreichen Inkompatibilitäten zwischen verschiedenen Paket-Versionen ist es hilfreich, die Miniconda-Umgebung zu sichern, sobald sie funktioniert.

IDE

Für das Schreiben des Codes und die Verwaltung der Dateien wurde die integrierte Entwicklungsumgebung (IDE) DataSpell von JetBrains verwendet. Die IDE bietet viele Funktionen

zur Datenverwaltung, zur Anzeige von CSV-Dateien und zur interaktiven Ausführung von Skripten. Darüber hinaus erleichtert sie die Arbeit mit Jupyter Notebooks, ohne dass eine Konsole genutzt werden muss. Es ist lediglich erforderlich, die Miniconda-Umgebung mit der IDE zu verbinden, um auf die installierten Pakete zugreifen zu können. JetBrains bezeichnet DataSpell als die IDE für professionelle Data Scientists.

Werkzeug zum Labeln/Annotieren

Für das Annotieren von Daten, insbesondere von Bildern, wurde Label Studio genutzt. Dieses Tool kann lokal auf dem eigenen Computer oder Server installiert werden und enthält eine Nutzerverwaltung in Form von Nutzerprofilen, um die Arbeit aufzuteilen. Nach der Installation kann auf das Tool über einen Browser zugegriffen werden. Hierzu tippt man die Adresse des Computers und einen vorgegebenen Port in die Adresszeile.
Da Label Studio eine Open-Source-Software ist, bietet es viele Erweiterungsmöglichkeiten. Ein weiteres Feature besteht darin, dass ein Machine-Learning-Backend angebunden werden kann, das ein vortrainiertes Modell enthält und die hochgeladenen Bilder vorannotiert. Dies erspart Zeit und ermöglicht später die Klassifizierung, Filterung und den Download ganzer Datensätze.

3.3 Transfer Learning und Modell

In diesem Projekt wurde in Tensorflow recherchiert und Transfer Learning angewendet. Es wurden verschiedene Modelle miteinander verglichen und dabei wurde der Schwerpunkt auf Genauigkeit und Geschwindigkeit gelegt. Durch den Einsatz von Transfer Learning konnten die Vorteile gegenüber dem Trainieren eines Modells "from scratch"genutzt werden. Somit wurden Zeit und Ressourcen gespart. TensorFlow bietet eine Liste an state-of-the-art vortrainierten Modellen, die auf dem COCO-2017 Datensatz traniert wurden.

In dem Projekt wurden die Modelle 'SSD MobileNet V2', 'SSD ResNet152 V1', 'Faster R-CNN ResNet101 V1' und 'CenterNet HourGlass104' zum Erkennen von Personen auf dem Datensatz angewendet. Danach wurden die Ergebnisse der unterschiedlichen Modelle verglichen und die Entscheidung fiel auf CenterNet, da dieses die höchste Genauigkeit, Geschwindigkeit und die optimalsten Bounding Boxen geliefert hat.

Parameter konfigurieren

Um ein vortrainiertes Modell weiter zu trainieren, kann man die Parameter in der Konfigurationsdatei anpassen. Die Konfigurationsdatei wird normalerweise vom Entwickler des vortrainierten Modells bereitgestellt und heißt "pipeline.config".
In der Konfigurationsdatei können verschiedene Parameter angepasst werden, um das Modell zu optimieren. Einige der wichtigsten Parameter sind die Anzahl der Steps, die Aktivierungsfunktion, die Skalierung des Input-Bildes, die Data Augmentation und die Pfade zu den Trainingsdaten.
Die Anzahl der Steps gibt an, wie oft das Modell während des Trainings einen einzelnen Datensatz (Batch) verarbeitet. Die Anzahl der Schritte pro Epoche kann mit der Formel (Steps pro Epoche = Anzahl der Bilder / Batchsize) berechnet werden. Daraus ergibt sich für die gegebenen Beispiele:
1 Epoche (3000 Bilder) = 3000 Steps (Batchsize = 1)
2 Epoche (6000 Bilder) = 6000 Steps (Batchsize = 2)
Mit einer Batchsize von 4 und einem Trainingsdatensatz von 3416 Bildern besteht eine Epoche in diesem Fall aus 854 Schritten.

Um das Modell zu trainieren, können verschiedene Batchsizes und Epochenanzahlen ausprobiert werden. Eine höhere Anzahl von Steps und Epochen kann möglicherweise zu einer höheren Genauigkeit des Modells führen. Für das gegebene Beispiel wurde das Modell mit insgesamt 34160 Schritten trainiert, was 40 Epochen entspricht.

3.4 Ergebnisse

Das Modell CenterNet wurde erfolgreich mit 3116 Bildern für 40 Epochen umtrainiert und anschließend mit 300 Bildern (100 pro Klasse: Mann, Kind, Frau) evaluiert. Das Erhöhen der Anzahl der Epochen führte nicht zu einer höheren Genauigkeit, sondern zu schlechteren Ergebnissen. Das trainierte Modell war in der Lage, Kinder, Männer und Frauen zu erkennen. Der Fokus lag auf den Ergebnissen der Erkennung von Kindern.

Insgesamt gab es über 1279 Bilder mit Kindern. Das Modell identifizierte 1652 vermeintliche Kinderbilder aus einer Gesamtzahl von 5484 Bildern mit unterschiedlichen Motiven. Von den manuell klassifizierten 1279 Bildern mit Kindern erkannte das Modell 1134 Bilder als solche. Trotzdem gab es Ungenauigkeiten. Das Modell klassifizierte 518 Bilder fälschlicherweise als Kind-Bilder, obwohl sie keine Kinder enthielten. Des Weiteren wurden 145 Bilder mit Kindern nicht erkannt.

Auf dem Bild 3.1 ist eine Gruppe von Männern zu sehen. Das Modell hat in fortschreitenden Epochen immer bessere Ergebnisse erzielt. Auf den ersten Blick ist das ein zufriedenstellendes Ergebnis. Jedoch zeigt sich beim Bild 3.2 ein Negativbeispiel.

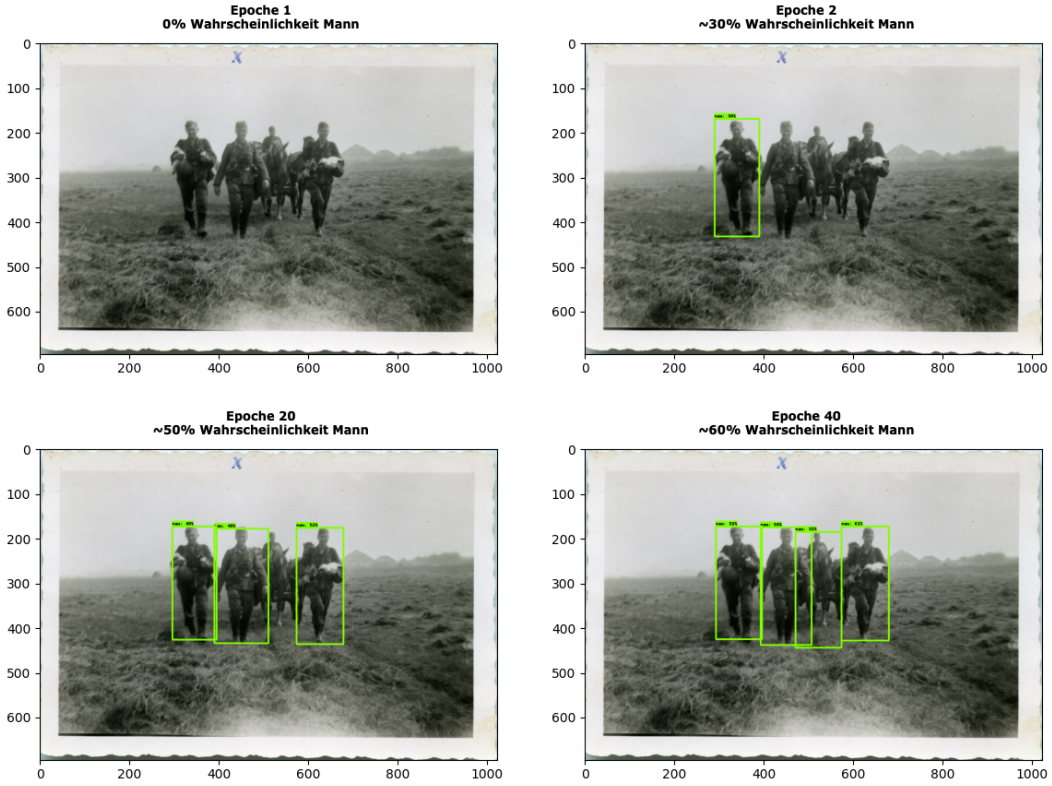

Abbildung 3.1: 'Gruppe von Soldaten', BA 2004-00383, Badisches Landesmuseum Karlsruhe

Hier hat das Modell bei zu wenig Epochen nichts erkannt. Leider war das Modell sich ab 20 Epochen sicher etwas erkannt zu haben. Jedoch ist sich das Modell nicht sicher, die Wahrscheinlichkeiten für 'man' und 'woman' sind beide zu hoch. Deswegen wurde die Ritterrüstung als Mann und Frau erkannt.

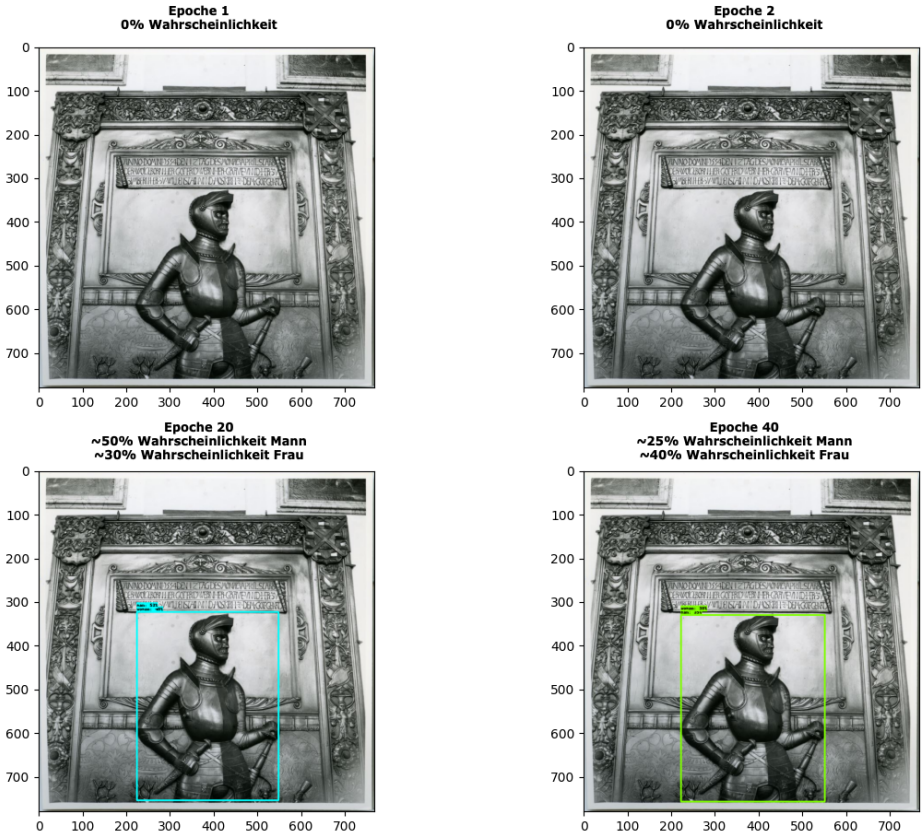

Abbildung 3.2: 'Ritter', BA 2004-00362, Badisches Landesmuseum Karlsruhe

3.4.1 Schlussfolgerung

Basierend auf den Ergebnissen der Evaluierung des trainierten CenterNet-Modells kann festgestellt werden, dass das Modell insgesamt in der Lage war, Kinder, Männer und Frauen zu erkennen. Insbesondere bei der Erkennung von Kindern gab es jedoch Ungenauigkeiten, wie die Identifizierung von Bildern ohne Kinder als solche und das Nichterkennen von Bildern mit Kindern. Obwohl das Erhöhen der Anzahl der Epochen nicht zu besseren Ergebnissen führte, hat das Modell in fortschreitenden Epochen dennoch verbesserte Ergebnisse erzielt. Zusammenfassend lässt sich sagen, dass das trainierte CenterNet-Modell ein vielversprechender Ansatz für die Erkennung von Personen in Bildern ist, jedoch noch Verbesserungspotential aufweist.

3.4.2 Reflektion

Es wurde ein Problem bei der Erkennung von Personen in verschiedenen Altersklassen festgestellt. Das Modell wurde mit den Klassen 'man', 'woman' und 'child' trainiert. Die Vermutung ist, dass das Modell es einfacher findet, Männer und Frauen zu unterscheiden.

Diese weisen oft größere Unterschiede in Merkmalen wie Gesichtsbehaarung, Kleidungsstil und körperlichen Merkmalen auf. Kinder hingegen tragen oft ähnliche Kleidung wie Erwachsene, nur in unterschiedlichen Größen. Zudem hängt das Aussehen von Kindern stark vom Alter ab. Hier wäre es sinnvoll gewesen, Kinder in verschiedene Altersgruppen wie 'infant', 'toddler', 'child' und 'teenager' zu unterteilen. Ihre körperlichen Merkmale unterscheiden sich in jeder Altersstufe oft stark. Zum Beispiel ist der Kopf bei Kindern oft größer im Verhältnis zum Rest des Körpers, wenn man sie mit Erwachsenen vergleicht.
Es wird zudem vermutet, dass die Genauigkeit des Modells mit mehr Trainingsdaten erhöht werden kann.

Literaturverzeichnis

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

Amit, Y., Felzenszwalb, P., and Girshick, R. (2020). Object detection. *Computer Vision: A Reference Guide*, pages 1–9.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.

Basha, S. S., Dubey, S. R., Pulabaigari, V., and Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378:112–119.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Harshit, Jin, K. H., Nguyen, H. Q., McCann, M. T., and Unser, Michael, G. (2018). Cnn-based projected gradient descent for consistent ct image reconstruction. *IEEE transactions on medical imaging*, 37(6):1440–1453.

Jampert, K., Zehnbauer, A., Best, P., Sens, A., Leuckefeld, K., and Laier, M. (2009). Kinder- sprache/n stärken. *Ministerium für Generationen Familien und Integration des Landes NRW (Hrsg.), Kinder bilden Sprache- Sprache bildet Kinder. Sprachentwicklung und Sprachförderung in Kindertagesstätten*, pages 21–32.

Jie, H. J. and Wanda, P. (2020). Runpool: A dynamic pooling layer for convolution neural network. *Int. J. Comput. Intell. Syst.*, 13(1):66–76.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kurbiel, T. and Khaleghian, S. (2017). Training of deep neural networks based on distance measures using rmsprop. *arXiv preprint arXiv:1708.01911*.

Law, H. and Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750.

Li, Z., Dong, M., Wen, S., Hu, X., Zhou, P., and Zeng, Z. (2019). Clu-cnns: Object detection for medical images. *Neurocomputing*, 350:53–59.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Narkhede, M. V., Bartakke, P. P., and Sutaone, M. S. (2022). A review on weight initialization strategies for neural networks. *Artificial intelligence review*, 55(1):291–322.

Phaisangittisagul, E. (2016). An analysis of the regularization between l2 and dropout in single hidden layer neural network. In *2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS)*, pages 174–179. IEEE.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Taylor, L. and Nitschke, G. (2018). Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*, pages 1542–1547. IEEE.

Wang, D., Devin, C., Cai, Q.-Z., Yu, F., and Darrell, T. (2019). Deep object-centric policies for autonomous driving. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8853–8859. IEEE.

Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*.

Abkürzungsverzeichnis

CNN Convolutional Neuronal Network. 3, 11

FE Feature Extraction. 13, 15

FPN Feature Pyramid Networks. 10

FT Fine Tuning. 13, 15

IC Image Classification. 8

OD Object Detection. i, 8, 9, 11, 17, 18

R-CNN Region Based Convolutional Neural Networks. 10

SGD Stochastic gradient descent. 6, 16

TL Transfer Learning. 12

Glossar

adam Adam steht für "Adaptive Moment Estimation" und ist ein Optimierungsalgorithmus für die Stochastic Gradient Descent (SGD) Methode in maschinellem Lernen. 5

backpropagation Backpropagation ist ein Algorithmus zur Optimierung von künstlichen neuronalen Netzen. 5

Label-Studio Ein Open-Source Tool zum annotieren von Bildern im Webbrowser. 18

Sigmoid-Funktion Eine Sigmoidfunktion, Schwanenhalsfunktion, Fermifunktion oder S-Funktion ist eine mathematische Funktion mit einem S-förmigen Graphen. 6