HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Johannes Krause

Der Weg zur Brückenklassifizierung

Generierung, Annotation und Klassifizierung eines Datensatzes

2019

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Der Weg zur Brückenklassifizierung

Generierung, Annotation und Klassifizierung eines Datensatzes

Autor:
Johannes Krause

Studiengang:
Medieninformatik & interaktives Entertainment

Seminargruppe:
MI15w3-b

Erstprüfer:
Prof. Dr. Thomas Haenselmann

Zweitprüfer:
Maik Benndorf

Mittweida, Januar 2019

Bibliografische Angaben

Krause, Johannes: Der Weg zur Brückenklassifizierung, Generierung, Annotation und Klassifizierung eines Datensatzes, 55 Seiten, 47 Abbildungen, 10 Tabellen, Anlage: DVD mit Quellcodes und Modellinformationen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2019

Satz: LaTeX
Druck:
Werkstätten für Buchbinderei Donath KG
Zietenstraße 65
09130 Chemnitz

Referat

In dieser Arbeit wird darauf eingegangen, wie zum Zwecke der Klassifizierung von Brückenbildern ein Datensatz erhoben, annotiert, augmentiert und verschiedene Modelle neuronaler Netze darauf trainiert wurden die Brückenbilder in eine von sieben Kategorien einzuordnen. Dabei wird kurz auf die übergeordnete Bemühung eingegangen, mithilfe eines solchen Netzes einen Beitrag zur Abmilderung der Auswirkungen katastrophaler Ereignisse zu bieten. Anschließend werden die vorgenommenen Schritte im Einzelnen erläutert, die zu den im letztem Kapitel gezeigten Ergebnissen führten. Abschließend werden die Ergebnisse diskutiert, Probleme und mögliche Fehlerquellen angesprochen und ein Ausblick auf Weiterführung der Bemühung zur korrekten Brückenklassifikation gegeben.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

CNN ..... Gefaltetes neuronales Netz, Seite 7

EER-Model ..... Enhanced entity-relationship-Model, Seite 20

FFN ..... Feed-Forward-Netzwerk, Seite 5

ReLU ..... Rectified linear unit, Seite 6

SGD ..... Stochastischer Gradientenabstieg, Seite 10

1 Einleitung

1.1 Motivation

Die Hochschule Mittweida forscht seit 2016 in Zusammenarbeit mit der Universität der Bundeswehr in München an Möglichkeiten und Methoden zur schnellen und kostengünstigen Einschätzung der Stabilität von Brücken nach katastrophalen Überflutungen. Ziel dieser Bemühungen besteht in der Entwicklung einer App, die den Benutzer darüber informiert, ob es sicher ist, eine bestimmte Brücke im Katastrophengebiet zu überqueren, und eine Empfehlung für die verbleibende maximale Tragekapazität angibt. Im Rahmen dieser Forschung wurde bereits gezeigt, dass es mit in der Bevölkerung verbreiteten Smartphones möglich ist, mithilfe deren eingebauter Beschleunigungssensor die veränderte Eigenfrequenz einer Brücke vor und nach einer Beschädigung zu messen [7]. Dieses Forschungsergebnis konnte in einer weiteren Arbeit bestätigt werden, in der die Integrität der Lesumbrücke in Bremen mithilfe dieser Methode beurteilt wurde. Diese Arbeit behandelte zudem die Problematik, wie geeignete Messpunkte auf einer potentiell einsturzgefährdeten Brücke erreicht werden können [8]. Noch ungelöste Probleme dieser Methode bestehen darin, für Laien geeignete Messpunkte zu finden, und gemessene Werte korrekt zu interpretieren. In Katastrophenfällen stehen für solche Aufgaben qualifizierte Spezialisten möglicherweise nicht, oder nicht rechtzeitig, zur Verfügung. Eine mögliche Lösung dieser Probleme besteht darin, vom selben Smartphone, mit dem die Frequenzmessung durchgeführt wird, ein Bild der betreffenden Brücke zu machen, und in der App unter Einsatz eines künstlichen neuronalen Netzes die natürliche Eigenfrequenz einer Brücke bestimmen zu lassen. Dadurch wird das Betreten der möglicherweise einsturzgefährdeten Brücke zur Bestimmung ihrer Stabilität obsolet.

Künstliche neuronale Netze werden für eine Vielzahl an Aufgaben in der Bildbearbeitung und -erkennung eingesetzt. Ihr Anwendungsbereich reicht von Gesichts- und Objekterkennung über Supersampling bis hin zur Videosynthetisierung [5] [13] [16]. Sie übertreffen klassische Algorithmen dabei mitunter bei weitem [12]. Ihr Potential war und ist Gegenstand weitreichender Untersuchungen da der Grund, warum sie viele Aufgaben so gut durchführen, noch weitestgehend unbekannt ist [18] [19].

Ähnlich ihrem biologischem Vorbild, dem menschlichem Gehirn, lernen künstliche neuronale Netze durch Beispiele. Das gleiche Netz kann dabei auf unterschiedliche Aufgaben trainiert werden. Zu diesem Zweck sind oft umfangreiche Datenmengen notwendig, um dem Netz ein größtmögliches Abstraktionsvermögen anzutrainieren. Um zur Brückeninspektion sinnvoll eingesetzt werden zu können ist es also zuvor notwendig, einen umfangreichen Datensatz an Training- und Testdaten zu generieren, und diese mit den zu ermittelnden Werten zu annotieren. Diese Arbeit beinhaltet den

Ablauf der Generierung und Annotation ebenjener Testdaten, und eine Untersuchung der Eignung verschiedener Architekturen von neuronalen Netzen zur Erkennung von Brückentypen.

1.2 Aufgabenstellung

Für diese Arbeit war folgende Aufgabenstellung gegeben:

Generierung Im ersten Schritt muss eine ausreichend große Datenmenge in Form von Abbildungen von Brücken erhoben werden. Aufgrund der Größe der Menge und der notwendigen Divergenz ist eine manuelle Erhebung ausgeschloßen. Die Abbildungen müßen eine Brücke gut sichtbar im Fokus beinhalten. Anforderungen an die Größe oder Auflösung der Bilder besteht nicht.

Annotation Um zum Training eines neuronalen Netzes verwendet werden zu können muss ein Bild mit einer zusätzlichen Information annotiert werden. Diese Information entspricht einer von sieben unterschiedlichen Brückenkategorien. Eine größere annotierte Datenmenge verspricht bessere Trainingsergebnisse. Eine Mehrzahl an Personen soll in der Lage sein, gleichzeitig die generierten Brückenbilder zu annotieren, ohne bereits bestehende Annotationen zu überschreiben. Zu diesem Zweck soll ein bestehendes Software-System auf Eignung zur schnellen und verteilten Bildannotation verwendet, oder eines entwickelt werden.

Klassifikation Die Kategegorie von möglichs vielen der nicht zum Training verwendeten annotierten Brückenbilder sollen von einem neuronalem Netz korrekt erkannt werden. Hierfür werden verschiedene Netzarchitekturen, Parametrisierungen und Bildermengen verwendet und die daraus entstehenden Ergebnisse miteinander vergleichen. Eine Zielgenauigkeit wurde hierfür nicht definiert.

1.3 Abgrenzung

Nicht Ziel dieser Arbeit ist es, das beste Netzmodell zur Klassifizierung von Brückenbildern zu finden. Neue Modellierungen und Architekturen neuronaler Netze zur Bilderkennung erscheinen regelmäßig, und es ist somit sehr schwierig, eine endgültige Architektur zu finden Ebenfalls ist es nicht Bestandteil dieser Arbeit die Leistung von künstlichen neuronalen Netzen mit anderen Algorithmen zur Bildererkennung und -klassifizierung zu vergleichen. Neuronale Netze haben ihre Überlegenheit im Gebiet der Mustererkennung bewiesen, weswegen auf einen weiteren Nachweis in dieser Arbeit verzichtet wird [12].

1.4 Übersicht

Diese Arbeit geht zuerst auf einige Grundlagen von neuronalen Netzen ein, die zum weiteren Verstehen der Arbeit notwendig sind. Das nächste Kapitel „Durchführung“ ist in mehrere Unterkapitel aufgeteilt: In „Generierung“ wird diskutiert, wie ein Datensatz an ausreichend umfangreichen Brückenbildern generiert wurde. Anschließend werden in „Annotation“ die Methoden und Werkzeuge vorgestellt, mit denen dieser Datensatz annotiert wurde, welche Kategorien für diese spezifiziert wurden und welche Mengen an Bildern die zu den jeweiligen Kategorien passen, bis zum Zeitpunkt des Verfassens dieser Arbeit erfasst wurden. Das entstandene Ungleichgewicht der Klassen und Strategien, mit diesem im weiteren Verlauf umzugehen, werden im darauf folgendem Unterkapitel „Augmentation“ vorgestellt. „Modelle“ zeigt detailliert den Aufbau jedes Modells, dass zum Training verwendet wurde. Im Unterkapitel „Training“ wird auf die Details des Trainingsvorgangs eingegangen, insbesondere die Auswirkungen unterschiedlicher Parametrisierung. Unter „Qualitätskriterien“ werden die Quantifizierungen von Merkmalen definiert, anhand der die Güte der Modelle bewertet wird. Im letztem Unterkapitel „Experimente“ werden die Ergebnisse vorgestellt, die mit verschiedenen Architekturen unter variierenden Parametern erzielt wurden, um die annotierten Brückenbilder korrekt zu klassifizieren, und erläutert, mit welchen Werkzeugen diese implementiert wurden. Abschließend werden die wichtigsten Erkenntnisse zusammengefasst und ein Ausblick auf notwendige Schritte für zukünftige Arbeiten gegeben.

2 Grundlagen

2.1 Künstliche neuronale Netze

Künstliche neuronale Netze wurden erstmalig 1943 von Warren McCulloch und Walter Pitts vorgestellt [9]. Sie können als gerichtete Graphen mit gewichteten Kanten beschrieben werden, die über eine bestimmte Anzahl an Quellen und Senken verfügen, und deren Knotenpunkte, Neuronen genannt, eine Aktivierungsfunktion haben. Neuronale Netze werden in zwei Typen aufgeteilt, Feed-Forward- und rekurrente Netze. Ein rekurrentes Netz zeichnet sich dadurch aus, dass es mindestens eine Schleife beinhaltet während Feed-Forward-Netzwerke (FFN) Signale ausschließlich vorwärtsgerichtet weiterleiten (Siehe Abbildung 2.1). Die erste und letzte Schicht von neuronalen Netzen werden Ein- und Ausgabeschicht genannt, dazwischen liegende Schichten als versteckte Schichten.

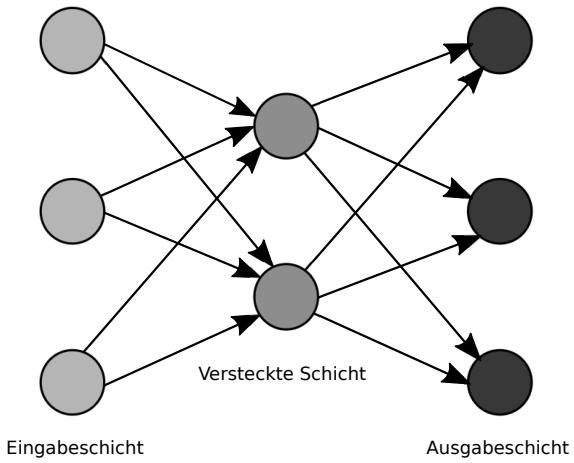


Abbildung 2.1: Beispiel: Feed-Forward-Netzwerk
Diese Abbildung stellt ein Feed-Forward-Netzwerk dar, in der alle Neuronen einer Schicht mit allen Neuronen der jeweils nächsten und vorhergehenden Schicht verbunden sind.

Neuronen der Eingabeschicht senden die ihnen eingegebenen Daten als Signal über ihre Kanten aus. Das mit dieser Kante verbundene Eingabeneuron der darauffolgenden Schicht empfängt dieses Signal als Produkt des Wertes des Ausgabeneurons und des Gewichtes der verbindenden Kante. Der Wert eines Neurons berechnet sich aus der Summe aller gewichteten Eingangssignale und seiner Aktivierungsfunktion. Der berechnete Wert stellt wiederrum das Ausgangssignal des Neurones dar, welches als Ausgabe weiter an verbundene Neuronen der nächsten Schicht propagiert wird. Der Wert eines Neurons n lässt sich damit folgendermaßen darstellen:

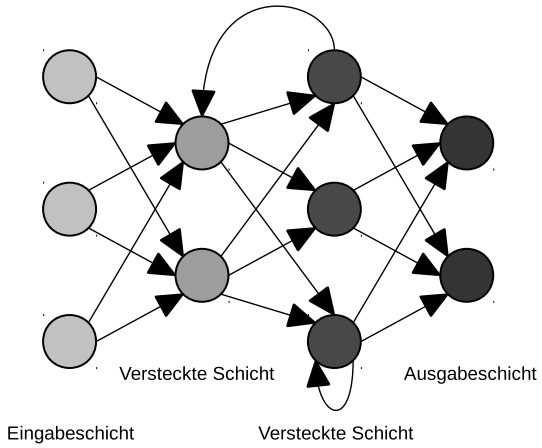


Abbildung 2.2: Beispiel: Rekurrentes Netzwerk
In einem rekurrentem Netzwerk kann ein Neuron eine gerichtete Kante auf ein Neuron einer früheren Schicht haben, als auch eine Kante zu sich selbst aufspannen.

$$n = \sigma(\sum_{i=1}^{N} v_i \omega_i)$$

wobei σ die Aktivierungsfunktion darstellt, i der Index eines Neurons aus einer vorrangegangenen Schicht, v der Wert dieses Neurones , ω die Gewichtung der verbindenden Kante und N die Anzahl der Neuronen aus der vorhergehenden Schicht. Aktivierungsfunktionen sind für alle Neuronen innerhalb einer Schicht gleich, Neuronen unterschiedlicher Schichten können jedoch andere Aktivierungsfunktionen haben. Eine Übersicht unter anderem häufig benutzter Funktionen findet sich in Tabelle 2.1.

	Formel
Sigmoid	$\frac{1}{1+e^{-x}}$
Hyperbolic Tangent	$\frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$
Rectified linear unit (ReLU)	$\begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Softmax	$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ wobei K die Anzahl Neuronen der Ausgabeschicht darstellt und z der Vektor an Eingangssignalen für die Neuronen der Ausgangsschicht.

Tabelle 2.1: Auflistung: Aktivierungsfunktionen

2.2 Perzeptron

Perzeptrone stellen FFN dar, dessen Neuronen mit allen Neuronen der direkt benachbarten Schichten miteinander verbunden ist. Abbildung 2.1 stellt ein Beispiel eines Perzeptrons dar.

2.3 Gefaltete neuronale Netze

Gefaltete neuronale Netze (CNN) eignen sich gut zur Bildererkennung. Sie gewinnen seit 2012 jährlich die stattfindende Large Scale Visual Recognition Challenge [12]. Sie stellen eine Unterklasse von FFN dar. Im Vergleich zu Perzeptronen gibt es jedoch wesentliche Unterschiede.

Aufgrund ihres Aufbaus können Neuronen innerhalb einer Schicht als Vektor betrachtet werden. In der Bilderkennung ist es aber dem Verständnis dienlicher, wenn die Schichten 2- oder 3-dimensional beschrieben werden können. Die Eingabeschicht bleibt auch in einem gefaltetem neuronalem Netz ein Vektor, kann aber abhängig von der Dimension eines Bildes 2-dimensionalen beschrieben werden. Bei einem Bild der Ausmaße $w \times h$ und dem daraus ergebendem Vektor

$$\begin{pmatrix} n_1 \\ n_2 \\ \dots \\ n_{wh} \end{pmatrix}$$

berechnet sich der Index für das entsprechende Neuron für ein Bildpunk (x,y) mittels

$$i(x,y) = x + w(y-1)$$

wobei $1 \leq x \leq w$ und $1 \leq y \leq h$.

Zudem sind nicht alle Neuronen zwischen den Schichten miteinander verbunden. Eine Faltungsschicht definiert stattdessen sogenannte Kernel. Ein Kernel ist eine quadratische Matrix der Größe $1 \leq m \leq min(w,h)$ mit derselben Anzahl Dimensionen wie die Eingabeschicht. Diese Kernel „wandern“ dabei über die Neuronen der Eingabeschicht in definerten Abständen, Stride genannt. Für jeden Stride verhalten sich die Felder eines Kernels wie die Kantengewichtungen zu einem Neuron der jeweils nächsten Schicht. Wird beispielsweise über ein Bild B der Dimensionalität $100x100$ ein Kernel K mit den Ausmaßen $5x5$ und einem Stride S von 1 definiert, entspricht dies einer $96x96$ großen Faltungsschicht F, auch Feature genannt, in der jedes Neuron mit jeweils $5x5$ Neuronen der vorangegangenen Schicht verbunden ist. Ein Stride von

5 hätte zur Folge, dass die Folgeschicht eine Dimensionalität von $20x20$ ausweisen müßte. Das Ausmaß einer Faltungsschicht $(w_F \times h_F)$ berechnet sich also aus der Größe der Bilddatenschicht $(w_B \times h_B)$, der Größe des Kernels $(w_K \times h_K)$ und dem Stride wie folgt

$$w_F = \frac{(w_B - w_K)}{S} + 1$$
$$h_F = \frac{(h_B - h_K)}{S} + 1$$

Die Neuronen des entstehenden Features berechnen sich folgendermaßen

$$F_{a,b} = \sum_{h=1}^{h_K} \sum_{w=1}^{w_K} (B_{c,d} * K_{w,h})$$

wobei sich die Indexvariablen c und d folgendermaßen berechnen

$$c = w + S(a-1)$$

$$d = h + S(b-1)$$

Abbildung 2.3 verdeutlicht dieses Verfahren noch einmal mit zweidimensionalen Matrizen für den Kernel, die Bilddaten und das entstehende Feature.

2.4 Max-Pooling

Unter Max-Pooling bezeichnet man innerhalb von CNN das Verfahren, Neuronen einer gefalteten Schicht zu reduzieren. Hierzu wird innerhalb eines gewissen Fensters die Neuronen mit dem größten Wert selektiert. Ähnlich eines Kernels wird für das MaxPooling ein Filter einer bestimmten Dimensionalität w_M, h_M definiert. Im Unterschied zu einem Kernel beinhaltet dieser Filter keine eigenen Werte, sondern stellt lediglich die Größe des Feldes, innerhalb der das Neuron mit dem größtem Wert selektiert wird, dar. Wie ein Kernel verfügt solch ein Filter auch über einen Stride. Abbildung 3.30 verdeutlicht dieses Verfahren noch einmal bildlich.

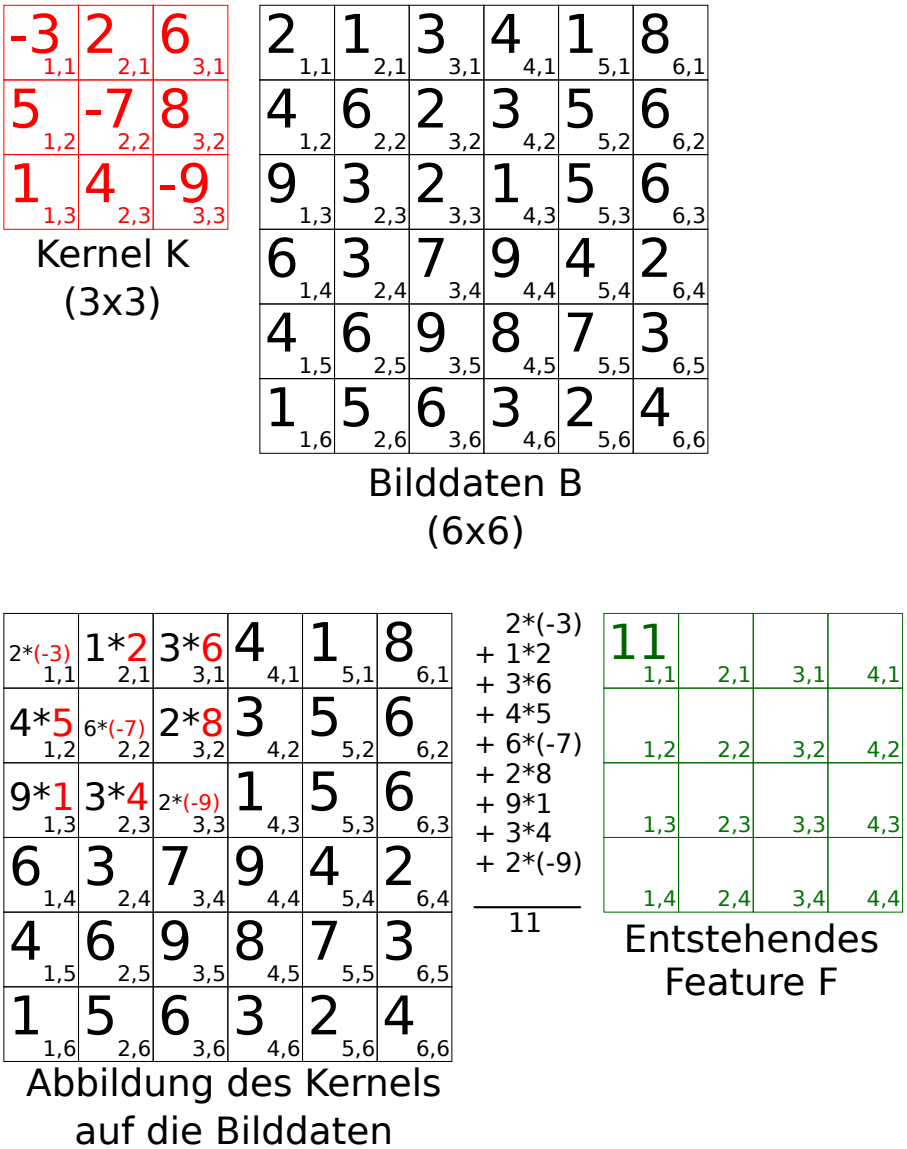


Abbildung 2.3: Veranschaulichung: Gefaltetes Netz

Berechnung eines Neurones eines Features aus 6x6 großen Bilddaten, einem Kernel von 3x3 und einem Stride von 1. Der nächste Schritt bestünde darin, das Feld (2,1) des Features zu berechnen, in dem der Kernel um die Größe des Strides, in diesem Falle also 1, nach Rechts verschoben wird. Dies wird wiederholt, bis alle Felder des Features berechnet worden sind.

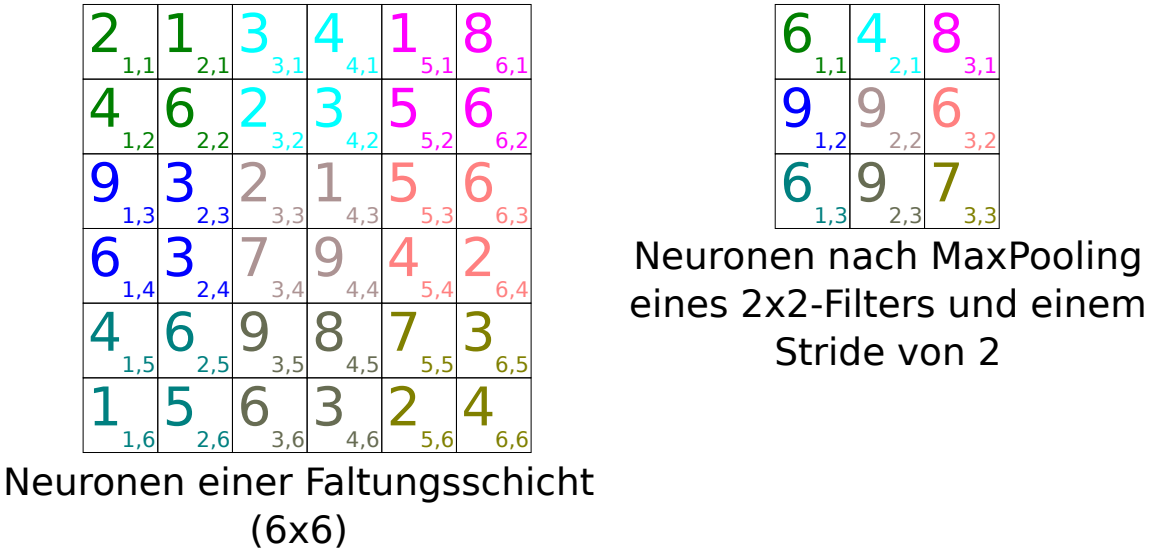


Abbildung 2.4: Veranschaulichung: MaxPooling

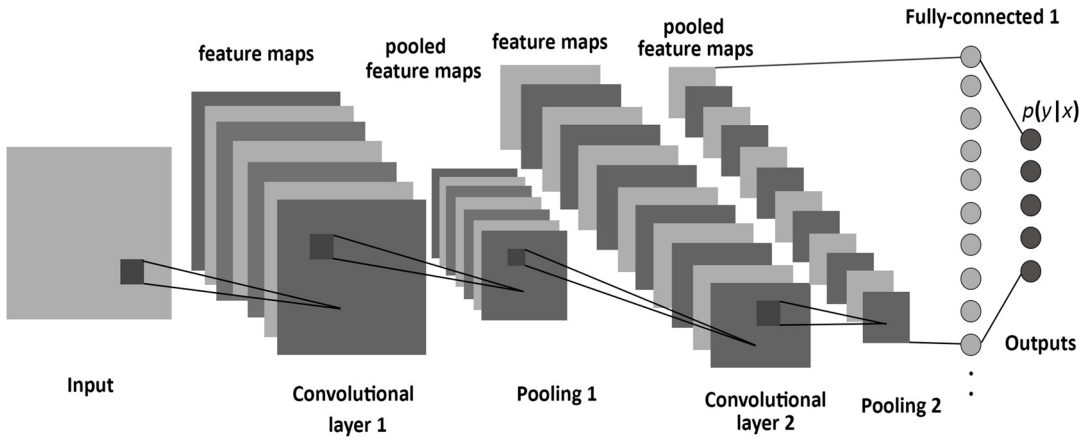


Abbildung 2.5: Beispiel: Gefaltetes neuronales Netz
Mit Faltungsschichten, MaxPooling und Feed-Forward-Schichten [1].

2.5 Training

Im Kontext neuronaler Netzwerke wird mit Training das Verfahren bezeichnet, bei der die Parameter innerhalb eines künstlichen neuronalen Netzes automatisch dahingehend angepasst werden, dass sie eine gewünschte Ausgabe produzieren. Am häufigsten werden dabei die Kantengewichte modifiziert. Ein verbreitetes Trainingsverfahren stellt dabei das überwachte Lernverfahren dar. In diesem Verfahren wird einem Netzwerk ein Vektor an Eingabedaten übergeben und die tatsächliche Ausgabe des Netzes mit einer gewünschten Ausgabe verglichen.

Mittels einer Kosten-Funktion $E(y,\hat{y})$, wobei y die gewünschte Ausgabe und $\hat{y}$ die tatsächliche Ausgabe ist, wird die Abweichung eines Ausgangsneurones ermittelt.

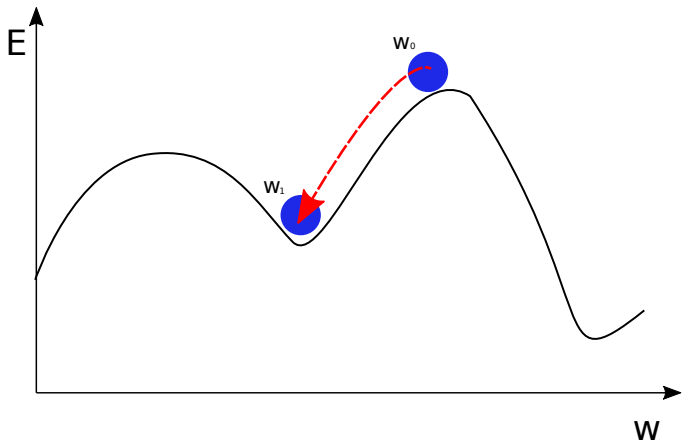


Abbildung 2.6: Veranschaulichung: Gradientenabstieg
w_0 stellt das Ausgangsgewicht einer Kantenverbindung dar. Diese Gewichtung wird mit jeder Iteration des stochastischen Gradientenabstieges angepasst, bis sie ein lokales Minimun w_1 erreicht. Man beachte, das mit einer höheren Kantengewichtung in diesem Beispiel ein geringerer Fehler produziert würde, der SGD aber aufgrund der Ausgangsposition von w_0 lediglich das lokale Minimum findet.

Die Kosten-Funktion kann hierbei beliebig gewählt werden, solange sie ableitbar ist. Anschließend werden die Kantengewichte über Backpropagation mittels eines ausgewählten Optimizer-Algorithmuses angepasst. Einer dieser Optimizer-Algorithmen ist der stochastische Gradientenabstieg (SGD), der die Kostenfunktion nach dem anzupassendem Kantengewicht ableitet:

$$\Delta w = \frac{\partial E}{\partial w}$$

$$w^{new} = w^{old} - \alpha \Delta w$$

wobei α den Lernkoeffizienten darstellt [6].

Ein Nachteil dieses Lernverfahrens besteht darin, dass durch Ermittlung des Abstieges lediglich lokale Minima der Kosten-Funktion gefunden werden können, es sei denn, das initiale Kantengewicht beginnt auf einem Abstieg zum globalem Minimum. Abhängig von den Ausgangswerten der Kantengewichte kann dasselbe neuronale Netz bessere oder schlechtere Ergebnisse liefern.

Ziel des Trainingsprozesses ist es, eine bestmögliche Generalisierungsfähigkeit des Netzwerkes zu erreichen. Das Netzwerk soll nicht nur die Daten aus den Trainingsdatensätzen korrekt klassifizieren können, sondern alle Daten des gleichen Types. Ein Netzwerk, dass auf die Erkennung von Brückentypen traininert wird, soll also explizit in der Lage sein, Brückenbilder korrekt zu klassifizieren, mit dem es zuvor nicht trainiert wurde.

	Formel
Mean Square Error	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Cross-Entropy	$-\frac{1}{n}\sum_{i=1}^{n}[y_i \ln \hat{y}_i] + (1-y_i)\ln(1-\hat{y}_i)]$ wobei n der Batch-Größe entspricht.

Tabelle 2.2: Vergleich: Kostenfunktionen

2.6 Overfitting

Als Overfitting (dt.: Überanpassung) wird der Verlust an Generalisierung eines Netzwerkes bezeichnet. Damit ist gemeint, dass ein Netzwerk zwar sehr gute Ergebnisse bei der richtige Klassifizierung der Datensätze erzielt, mit dem es trainiert wurde, die Klassifizierungsergebnisse trainingsfremder Datensätze jedoch darunter leiden. Overfitting stellt also das genaue Gegenteil der Zielstellung des Trainingsprozesses dar. Innerhalb eines Netzwerkes kann es während des Trainingsprozesses zu Overfitting führen, wenn das Netz uber zu viele und zu große versteckte Schichten verfügt. Overfitting kann vermieden werden in dem das Modell kleiner gestaltet wird, oder der Trainingsprozess gestoppt wird, bevor Overfitting einsetzt [4]. Eine weitere Möglichkeit zur Vermeidung von Overfitting sind Dropouts.

2.7 Dropout

Dropouts werden während des Trainingsprozesses auf eine einzelne Schicht angewendet. Während jeder Trainingsiteration werden Neuronen einer Schicht, auf die ein Dropout angewendet wird, mit einer Wahrscheinlichkeit $0 < p < 1$ für diese Iteration ignoriert. Diese Neuronen empfangen keine Signale mehr, noch senden sie ein Signal an Neuronen der nächsten Schicht. Die entsprechenden Kantengewichte werden beim Backpropagating ebenfalls ignoriert. Bei der nächsten Iteration werden wieder alle Neuronen zufällig für ein Dropout selektiert. Dieses Verfahren hat sich als effektiv zur Vermeidung von Overfitting erwiesen [15].

2.8 Epochen

Die Anzahl, mit der das Netzwerk mit den vollständigem Trainingsdaten lernt, bezeichnet man als Epochen. Im Laufe des Trainings mit verschiedenen Netzmodellen hat sich herausgestellt, dass eine höhere Anzahl Epochen zu verbesserte Klassifikationsergebnissen führt.

2.9 Batch

Batchgrößen definieren die Anzahl an Samples, für die der Abstieg des Netzwerk berechnet wird. Hierbei wird zun!achst der Abstieg für jedes Sample einzeln berechnet. Von diesen Werten wird anschließemd der Mittelwert berechnet, welcher dann über Backpropagating verwendet wird, die Kantengewichte anzupassen.

3 Durchführung

3.1 Generierung

Bei der Auswahl der Bilderdaten wurde besonders auf eine zukünftige Veröffentlichung des Datensatzes Rücksicht genommen. Aus diesem Grund kamen weitläufig verfügbare Bilderquellen wie beispielsweise die Google-Bildersuche nicht in Frage. Eine große Bilderdatenbank, welche über einen umfangreichen Satz an unter der CC-Lizenz veröffentlichten Bildern enthält, ist die Plattform `http://www.geograph.org.uk/`, eine Geographing-Plattform, welche Großbritannien und Irland in 'squares' aufteilt, und seine Nutzer dazu einlädt, möglichst viele Fotografien innerhalb dieser squares einzureichen und zu beschreiben.

`geograph.org.uk` verfügt über eine API und umfangreichen Data Dumps, welche zwar nicht die aktuellen Bilddaten enthalten, aber umfangreicher Metadaten. Innerhalb dieser Metadaten wurde nach dem Vorkommen der Zeichenfolge 'bridge' gesucht, und die extrahierten IDs in eine Datei gespeichert. Mithilfe der Details-API von geograph.org.uk wurde daraufhin den URI des dazugehörigen Bildes extrahiert und anschließend heruntergeladen. Mittels dieser Methode entstand ein, noch nicht annotierter, Datensatz von 151.072 Bildern.

3.2 Annotation

Im Rahmen dieser Bacheloararbeit werden die Brücken einer von sieben Kategorien unterteilt, wobei eine achte Kategorie für alle Bilder reserviert wurde, die sich in keine der anderen sieben Kategorien zuordnen ließen. Diese sieben Kategorien entspringen den sieben Strukturtypen, wie sie in einem Artikel über Brücken in der englischen Wikipedia beschrieben sind [17].

Zur Annotation wurde eine eigene Softwarelösung „Beam-Bridge-Inn“ entwickelt, die ein schnelles, verteiltes und kostenfreies Kategorisieren der erhobenen Bilderdaten mithilfe der Tastatur ermöglicht. Nach sorgfältiger Evaluation wurde dieser Schritt vorgenommen, da keine andere Annotationssoftware die für diese Arbeit gestellten Kriterien in ausreichender Hinsicht befriedigten, siehe Tabelle 3.1. „Beam-Bridge-Inn“ wurde als Plugin für das Content Management System OctoberCMS entwickelt und kann in dieses leicht integriert werden. „Beam-Bridge-Inn“ benötigt das Benutzer-Plugin von RainLab um ordnungsgemäß zu funktionieren.

Von Roger Kidd [20]. Im Rahmen dieser Arbeit werden Brücken als Balkenbrücken klassifiziert, wenn der Überbau vom Unterbau von einer geraden Kante der Brücke getrennt wird, und nur von den Lagern am Unterbau getragen wird. Balkenbrücken wurden dem Label ’1’ zugewiesen.

Abbildung 3.1: Abbildung: Balkenbrücke

Von N. Chadwick [23]. Als Gitterbrücken wurden jene Brücken klassifiziert, welche am Oberbau eine sichtbare Verstrebung von Pfeilern oder Stäben aufweisen, die an ihren Kreuzungspunkten miteinander verschmelzen. Gitterbrücken wurden dem Label ’2’ zugewiesen.

Abbildung 3.2: Abbildung: Gitterbrücke

Von Philip Halling [21]. Freitragende Brücken zeichnen sich durch das Fehlen von Lagern oder anderen Stützbauten aus. Diese Brücken bzw. Teile von ihnen weißen lediglich einen Stützpunkt mit dem Boden auf, welche die Last trägt. Freitragende Brücken wurden dem Label ’3’ zugewiesen.

Abbildung 3.3: Abbildung: Freitragende Brücke

Von Bill Nicholls [26]. Als Bogenbrücken wurden Brücken klassifiziert, die einen gut sichtbaren Bogen zwischen den Lagern der Brücke aufspannt, die oft als Durchgänge genutzt werden können. Bogenbrücken erhielten das Label ’4’.

Abbildung 3.4: Abbildung: Bogenbrücke

Von Robert Bone [24]. Langersche Balkenbrücken haben große Ähnlichkeit mit Balkenbrücken, mit dem Unterschied, dass ein Druckbogen auf beiden Seiten aufgespannt ist, welche gewöhnlich von beiden Seiten der Brücke oder ihren Lagern reicht. Die Balkenbrücke ist mit diesem Druckbogen durch Seile oder Stahlverstrebungen verbunden. Langersche Balkenrücken erhielten das Label ’5’.

Abbildung 3.5: Abbildung: Langersche Balkenbrücke

Von Fred Roberts [25]. Eine weitverbreitete Brückenart, die Hängebrücke, erkennt man an den an beiden Seiten errichteten Pylonen, zwischen denen ein Seil aufgespannt ist, an denen man die zu errichtende Passage aufhängt. Ihr wurde das Label ’6’ zugewiesen.

Abbildung 3.6: Abbildung: Hängebrücke

Von David Anstiss [22]. Schrägseilbrücken weisen Ähnlichkeit mit Hängebrücken in der Hinsicht auf, dass sie ebenfalls über Pylonen verfügen, die mithilfe von Lastseilen der Brücke eine zusätzliche Lastaufnahme ermöglichen. Im Unterschied zur Hängebrücke ist das Seil hierbei nicht zwischen den Pylonen aufgespannt, sondern an verschiedenen Stellen mit den zu stützendem Element direkt verbunden. Ihr wurde das Label ’7’ zugewiesen.

Abbildung 3.7: Abbildung: Schrägseilbrücke

Von Phil Williams [28]. Bilder, auf denen die Merkmale einer der sieben Brückentypen nicht erkannt werden können, werden als Unkategorisierbar eingestuft. Unkategorisierbare Brücken erhalten das Label ’8’.

Abbildung 3.8: Abbildung: Unkategorisierbar

	LabelD	DataTurks	Simple Image Annotator
Verteilt	In beschriebenem Funktionsumfang enthalten, aber nicht lauffähig.	Ja	Nein
Tastaturbedienung	Annotation erfolgt durch Selektion eines Bildbereiches und Einfügen von Freitext. Keine Kategorisierung. Mausbedienung zwingend nötig.	Ja	Nein. Mausbedienung zwingend nötig
Kostenfrei	Ja	Nur für Datensätze bis zu 100MB	Ja

Tabelle 3.1: Vergleich: Annotationssoftware

Insbesondere die Möglichkeit der Annotation ausschließlich mithilfe der Tastatur ermöglichte es, rasch einen annotierten Datensatz im Umfang von 10030 Bildern zu generieren. Diese 10030 annotierten Bilder teilen sich hierbei folgendermaßen in die verschiedene Kategorien auf:

	Anzahl	Davon Testdaten
Balkenbrücke	4612	923
Gitterbrücke	465	93
Freitragende Brücke	83	17
Bogenbrücke	4328	866
Langersche Balkenbrücke	207	42
Hängebrücke	220	44
Schrägseilbrücke	115	23
Gesamt	10030	2008

Tabelle 3.2: Aufzählung: Brückenbilder nach Kategorie

Der vollständige Datensatz wurde auf die Online-Plattform Kaggle hochgeladen: https://www.kaggle.com/johanneskrause/beambridgeinn

3.3 Beam-Bridge-Inn

Aufgrund mangelhafter Eignung frei verfügbarer Werkzeuge zur schnellen, verteilten Annotation großer Mengen an Bilddaten wurde ein Plugin für das Content-Management-System OctoberCMS entwickelt. Aufgrund seiner initialen Verwendung zur Annotation von Brückenbildern wurde diese Anwendungssoftware „Beam-Bridge-Inn “ getauft. Die Software lässt sich jedoch auch zur Annotation anderer Klassen von Bildern einsetzen.

Beim Design wurden insbesondere auf die in 3.1 genannten Qualitätskriterien geachtet: Bilder sollen verteilt, schnell über Tastatureingabe und ohne kommerzielle Angebote Dritter realisiert werden köennen. Zur Realisierung dieser Anforderungen wurde das relationale Datenbanksystem MySQL, das Content-Management-System OctoberCMS und das Benutzerverwaltungs-Plugin „User “ von Rainlab verwendet.

Aus den Anforderungen wurde ein Enhanced entity-relationship Model (EER-Model) wie in 3.9 abgebildet erstellt. Die einzelnen Datentabellen und -felder erfüllen hierbei folgende Aufgaben:

- **johanneskrause_bridgeannotator_bridge_types:** Beinhaltet die Labels, mit der Bilder annotiert werden können
  - *name:* Bezeichnung des Labels
  - *description:* Nähere Beschreibung des Labels
  - *hotkey:* Zeichencode des diesem Label zugewiesenem Hotkeys
- **johanneskrause_bridgeannotator_bridge_pictures:** Beinhaltet die Brückenbilder als auch deren Zugehörigkeit zu einem Arbeitspaket und deren annotierten Label
  - *grid_image_id:* Der Dateiname der Bilddatei
  - *work_package_id:* Das dem Bild zugewiesene Arbeitspaket
  - *bridge_type_id:* Das Label, dass dem Bild annotiert wurde
- **johanneskrause_bridgeannotator_work_package**
  - *user_id:* Der Benutzer, für den das Arbeitspaket reserviert wurde
  - *done:* Ein Flag, dass gesetzt wird, wenn alle Bilder innerhalb des Arbeitspaketes einem Brückentyp zugeordnet wurden
- **users:** Diese Datentabelle wurde vom „User “ Plugin von Rainlab bereitgestellt und dafür verwendet, registrierte Benutzer Datenpakete zuweisen zu können

Der vollständige Quellcode und eine Installationsanleitung finden sich in Anhang A und wurden auf ein Github-Repository hochgeladen: https://github.com/jkrause1/TheBeamBridgeInn

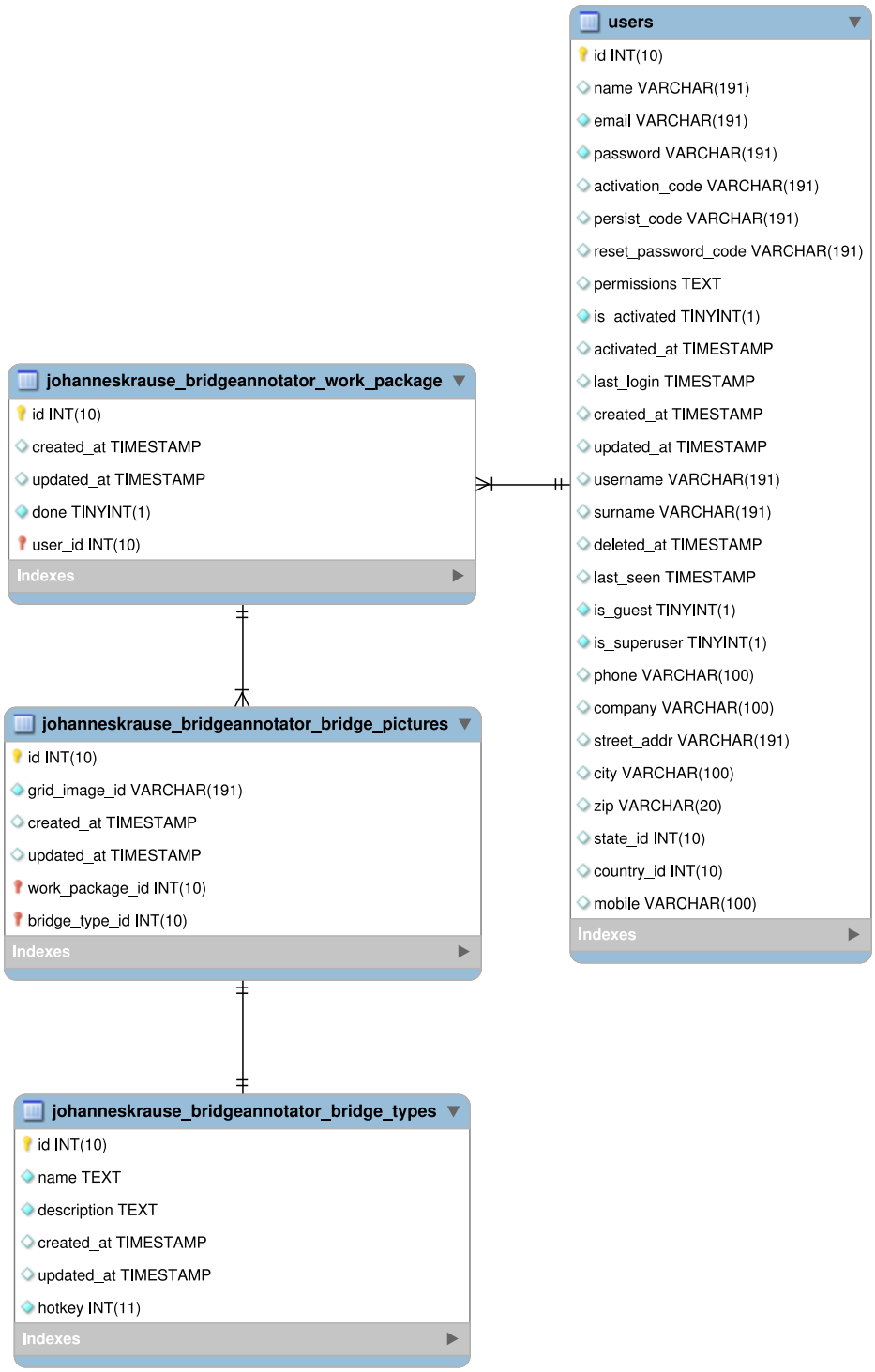


Abbildung 3.9: Beam-Bridge-Inn: Entity-Relationship-Model

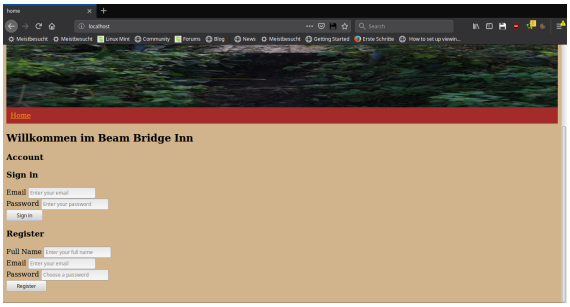


Abbildung 3.10: Screenshot: Login/Registrierungs-Maske des Beam-Bridge-Inn

Die Startseite besteht aus einer Maske zum Registrieren und zum Login bereits registrierter Benutzer. Welcher Aktivierungsmechanismus für den Registrierungsvorgang verwendet wird, hängt von der in OctoberCMS vorgenommenen Einstellungen ab.

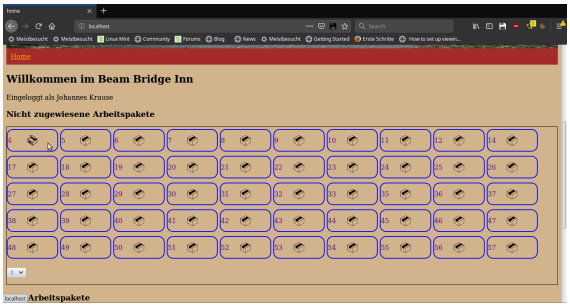


Abbildung 3.11: Screenshot: Reservierung eines Arbeitspaketes

Vor der Bearbeitung werden Arbeitspakete vom Nutzer reserviert. Dies stellt sicher, dass die darin enthaltenen Bilder exklusiv von diesem Benutzer annotiert werden.

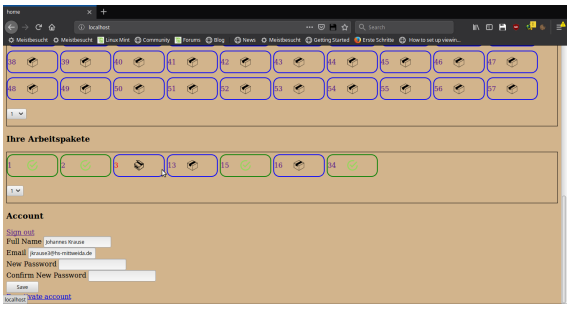
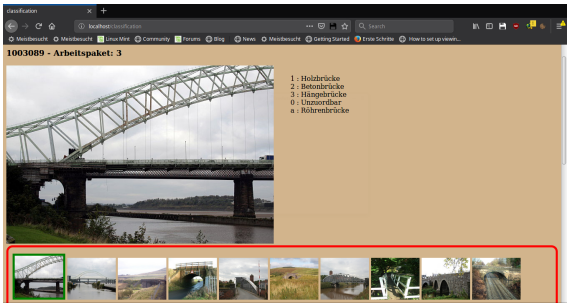


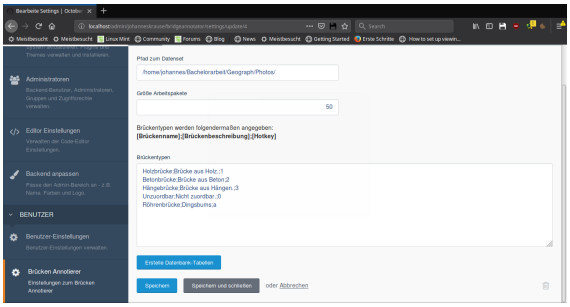
Abbildung 3.12: Screenshot: Auswahl eines Arbeitspaketes

Ein zuvor reserviertes Arbeitspaket kann nun zur Bearbeitung ausgewählt werden. Ein grüner Haken signalisiert ein Paket, in dem jedes Bild annotiert wurde. Es kann dennoch ausgewählt, und die darin enthaltenen Bilder reannotiert werden.



In der Annotationsmaske können die Bilder eines Arbeitspaketes mithilfe der angezeigten Hotkeys annotiert werden. Die Maske springt automatisch zum nächsten, nicht annotiertem Bild. Bereits annotierte Bilder können selektiert, und reannotiert werden.

Abbildung 3.13: Screenshot: Annotationsmaske



Nach der Installation des Beam-Bridge-Inn-Plugins wird der Einstellungsübersicht des OctoberCMS ein Eintrag „Brücken Annotierer “ hinzugefügt. Hier werden die zu annotierenden Bilder, die Größe der Arbeitspakete sowie die Labels und ihre Hotkeys eingestellt.

Abbildung 3.14: Screenshot: Administrationsoberfläche des Beam-Bridge-Inn

	Beschreibung
Menge A	Beinhaltete kein Ausgleich der Bilderdaten. Das Netz wurde mit dem Ungleichgewicht trainiert.
Menge B	Die Anzahl augmentierter Bilder pro Klasse berechnete sich aus der Differenz der aktuellen Anzahl an Bilder der Klasse und der Anzahl Bilder der am häufigsten vertretenen Klassen.
Menge C	Die Bilder der Klassen wurden unabhängig von ihrer Anzahl so häufig augmentiert, bis jede Klasse über eine Anzahl von 5000 Bildern verfügte.
Menge D	Die Bilderanzahl wird durch Augmentation verdreifacht. Jedes Bild wird hierbei exakt 2 mal augmentiert.
Menge E	Die Bilderanzahl aller Klassen wurden auf die Menge der am geringsten vorkommenden Klasse reduziert.
Menge F	Diese Menge besteht nur aus den 2 am häufigsten vertretenen Bilderklassen.

Tabelle 3.3: Auflistungen: Datenmengen Brückenbilder

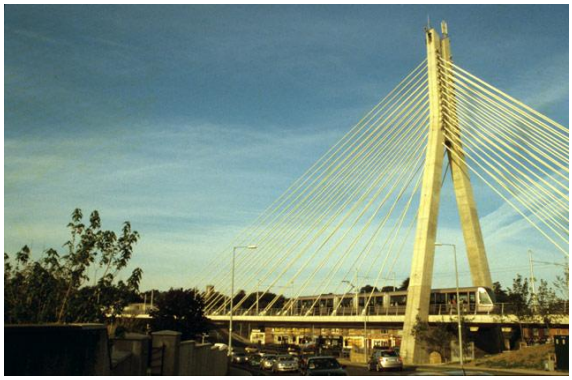
3.4 Augmentation

Nach der Generierung des Datensatzes stellte sich ein erhebliches Ungleichgewicht der Klasse heraus, wie in Tabelle 3.2 zu sehen. Balken- sowie Bogenbrücken sind quantitativ wesentlich häufiger vertreten als alle anderen Klassen. Um den Einfluss unterschiedlicher Mengen an Klassen während des Trainings zu untersuchen, wurden sechs verschiedene Datenmengen für das Netztraining erstellt (Siehe Tabelle 3.3).

Die Modelle wurden anschließend sowohl mit dem originalem, unverändertem Datensatz sowie dem augmentiertem Datensatz trainiert. Eine Augmentation des Testdatensatzes wurde nicht vorgenommen.

Ein zur Augmentation selektiertes Bild ging eine „Pipeline“ an Augmentation durch. Jede Augmentation konnte nur mit einer gewissen Wahrscheinlichkeit auftreten, und hat das Bild innerhalb des Rahmens angegebener Parameter zufällig augmentiert. Im Folgenden ist diese Pipeline aufgelistet.

Abbildung 3.15: Augmentation: Originalbild von John Armitstead [27].



Horizontaler Flip
Das Bild wird mit einer Wahrscheinlichkeit von $a = 0,5$ horizontal geflippt.

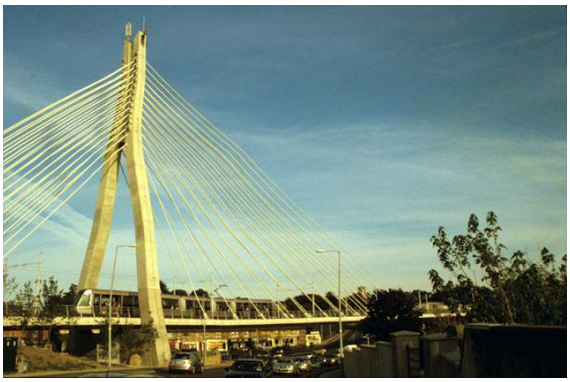
Abbildung 3.16: Augmentation: Flip

Rotation
Das Bild wird mit einem zufälligem Wert zwischen $min = -10$ und $max = 10$ Grad und einer Wahrscheinlichkeit von $a = 0,7$ im Uhrzeigersinn gedreht.

Abbildung 3.17: Augmentation: Rotation

Zoom
Mit einer Wahrscheinlichkeit von $a = 0,5$ wird mit einem zufälligem Faktor zwischen $min = 1,1$ und $max1,5$ in die Mitte des Bildes hineingezoomt.

Abbildung 3.18: Augmentation: Zoom



Verzerrung
Mit einer Wahrscheinlichkeit von $a = 0,9$ wird das Bild mit einem Ausmaß von $mag = 4$ verzerrt.

Abbildung 3.19: Augmentation: Verzerrung

Schnitt
Mit einer Wahrscheinlichkeit von $a = 0,7$ wird die Bildfläche auf den Wert $area = 0,6$ reduziert.

Abbildung 3.20: Augmentation: Schnitt

Farbsättigung
Die Farbsättigung wird mit einer Wahrscheinlichkeit von $a = 1$ zufällig zwischen einem Wert von $min = 0$ und $max = 1.5$ gesetzt.

$$a = 1, min = 0.0, max = 1.5$$

Abbildung 3.21: Augmentation: Farbsättigung

Helligkeit
Die Helligkeit wird mit einer Wahrscheinlichkeit von $a = 1$ auf einen zufälligen Wert zwischen $min = 0,1$ und $max = 1.5$ gesetzt.

Abbildung 3.22: Augmentation: Helligkeit

Kontrast
Der Kontrast des Bildes wird mit einer Wahrscheinlichkeit von $a = 1$ zufällig auf einen Wert zwischen $min = 0,1$ und $max = 1.5$ gesetzt.

Abbildung 3.23: Augmentation: Kontrast

Schrägstellung
Das Bild wird mit einer Wahrscheinlichkeit von $a = 0,7$ schräg gestellt.

Abbildung 3.24: Augmentation: Schrägstellung

Schreung
Das Bild wird mit einer Wahrscheinlichkeit von $a = 0,5$ mit einem zufälligem Wert zwischen $min = -25$ und $max = 25$ in Grad im Uhrzeigersinn geschert.

Abbildung 3.25: Augmentation: Scherung

Zur Anwendung der Augmentation wurde ein Python-Script geschrieben, welches das Augmentor-Modul verwendet [3]. Das Script lässt sich im Anhang A finden, sowie auf dem Github-Repository:
https://github.com/jkrause1/BridgeImageAugmentor

3.5 Modelle

Zur Klassifikation der erhobenen Bilderdaten wurden neun verschiedene Modelle entwickelt, auf die im folgendem Kapitel näher eingegangen werden soll. Alle Netzwerke wurden durch in mehrere Felder aufgeteilte Rechtecke, die Schichten, die durch Pfeile miteinander verbunden sind, dargestellt. Im linken Feld steht der Name und der Typ der Schicht. In den rechten Feldern jeweils das Ein- und Ausgangsformat der Signale. Das Eingangsformat einer Schicht und das Ausgangsformat seiner vorrangegangenen Schicht sind stets identisch. Das Eingangsformat der ersten Schicht beschreibt die Eingabeschicht und das Ausgangsformat der letzten Schicht die Ausgabeschicht. Siehe auch Abbildung 3.26,

Modelliert wurden Perzeptronen, CNN und eine modifizierte Variante das 2012 erstmals in der ImageNet Large Scale Visual Recognition Challenge verwendete, gefaltete Netzwerk AlexNet [10]. Die hier verwendete Version des AlexNet unterscheidet sich von dem im Original-Paper verwendetem Netzwerk darin, nur eine GPU zum Training und Klassifizieren zu verwenden. Das ursprüngliche AlexNet trainierte für die obere und untere Bildhälfte die Kernels zuerst separat mit jeweils einer GPU, bevor die extrahierten Features in der dritten Schicht der jeweils anderen GPU hinzugefügt wurden, woraufhin diese ihre Kernel wieder separat voneinander trainierten. Auf dieses Training separater Kernels und Zusammenführung extrahierter Features wurde hier verzichtet.

Es sei darauf hingewiesen, dass in allen vorliegenden Modellen die Eingabeschicht drei-dimensional mit den Ausmaßen von $100x100x3$ ausgegeben ist. In den Experimenten wurden mitunter andere Größen für die Eingabeschicht gewählt. Aus Gründen der Redundanzvermeidung werden diese hier nicht extra aufgeführt.

Bei allen Modellen wurde Cross-Entropy als Kostenfunktion E verwendet (Siehe Tabelle 2.2). Alle Schichten verwenden die ReLU-Aktivierungsfunktion, mit Ausnahme der Ausgabeschichten, welche Softmax verwenden (Siehe Tabelle 2.1).

(Schichtname): (Schichttyp)	input:	(Eingangsformat)
	output:	(Ausgangsformat)

Abbildung 3.26: Netzwerkschicht: Schichtbeispiel

flatten: Flatten	input:	(None, 1, 1, 256)
	output:	(None, 256)

Abbildung 3.27: Netzwerkschicht: Abflachungsschicht

Abflachungsschicht
Abflachungsschichten reduzieren die Dimensionalität des Eingabevektors. Der Ausgabevektor wird aus dem Produkt der Elemente des Eingabevektors gebildet. $y = e_a e_b e_c$

dense: Dense	input:	(None, 256)
	output:	(None, 4096)

Abbildung 3.28: Netzwerkschicht: Dichte Schicht

Dichte Schicht Dichte Schichten verknüpfen alle Neuronen der vorangegangenen Schicht mit den Neuronen ihrer Schicht.

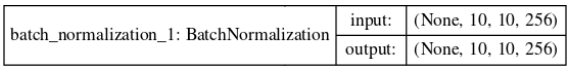


Abbildung 3.29: Netzwerkschicht: Batch-Normalisierungsschicht

Batch-Normalisierungsschicht In einer Normalisierungsschicht werden die Werte der Neuronen auf den Wertebereich $0 < x < 1$ normalisiert. Normalisierung von Neuronwerten hilft nachweislich dem neuronalem Netzwerk, schneller und besser zu lernen [14]

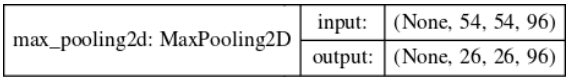


Abbildung 3.30: Netzwerkschicht: Max-Pooling-Schicht

Max-Pooling-Schicht Hierbei handelt es sich um eine Max-Pooling-Schicht wie in Sektion 2.4 erläutert.

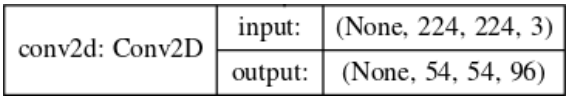


Abbildung 3.31: Netzwerkschicht: Faltungsschicht

Faltungsschicht Hierbei handelt es sich um eine Faltungsschicht wie in Sektion 2.3 erläutert.

3.5.1 1-Layer (1l_dense)

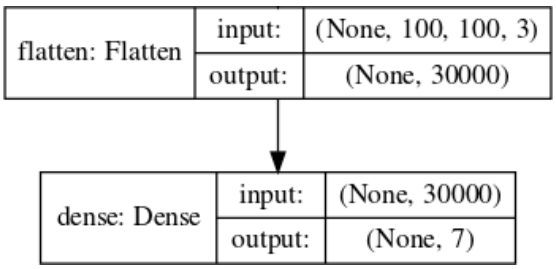


Abbildung 3.32: Netzmodell: 1-Layer Modell

Das 1-Layer-Modell besteht nur einer Abflachungs- und einer dichten Schicht. Es handelt sich also um ein einlagiges Perzeptron, welche sich primär dafür eignen, Datenmengen linear zu trennen.

3.5.2 Gefaltetes Netz (cnn)

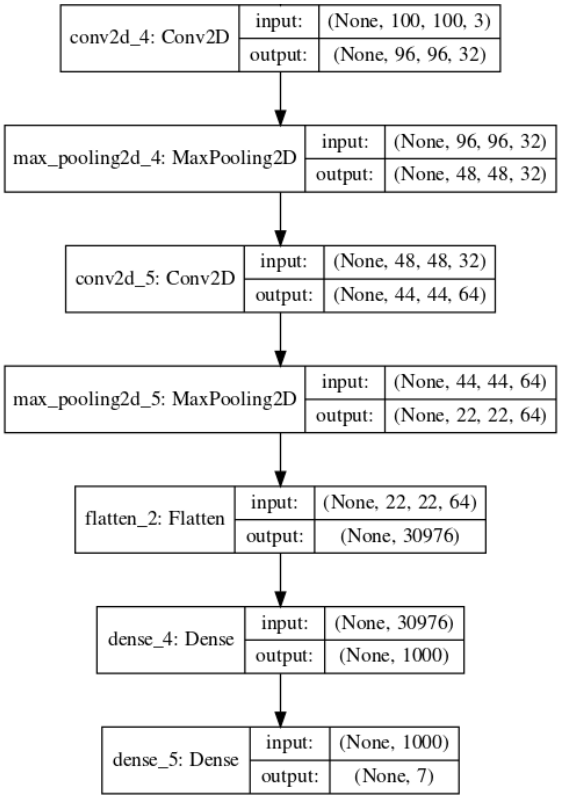


Abbildung 3.33: Netzmodell: Gefaltetes Netz

Ein einfaches CNN. Beide Faltungsschichten verwenden einen Kernel im Ausmaß von (5, 5) und einem Stride von (1, 1). Die erste Faltungsschicht verwendet 32 Kernel, die zweite 64. Die Max-Pooling-Schichten haben ein Ausmaß und einen Stride von (2, 2).

3.5.3 Erweitertes Gefaltetes Netz (cnn_extended)

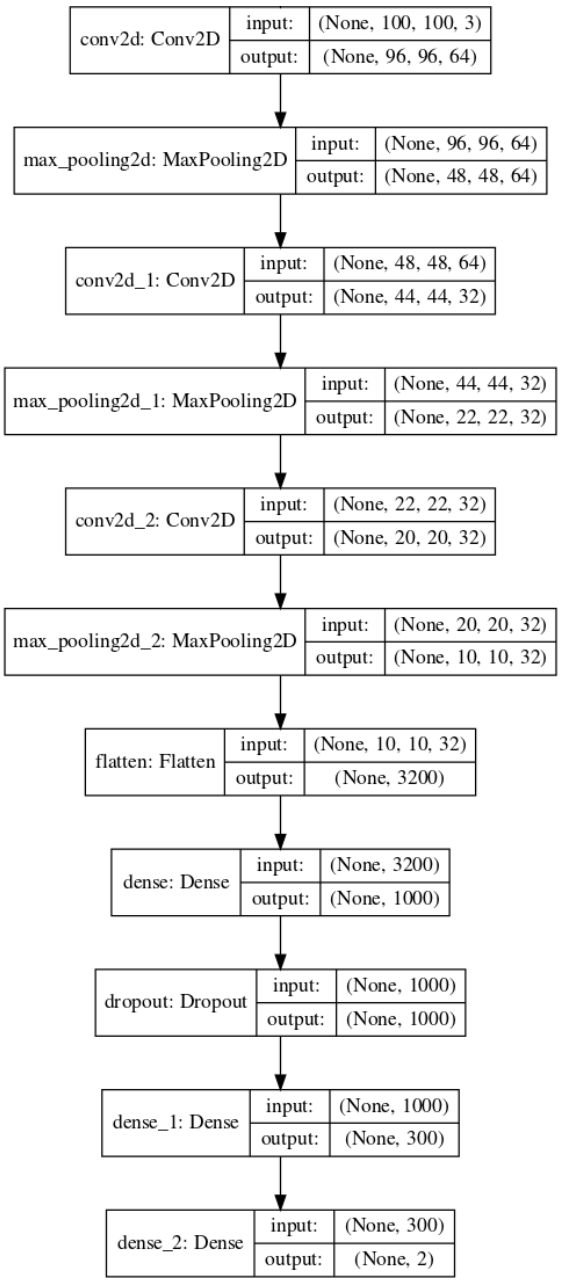


Abbildung 3.34: Netzmodell: Erweitertes gefaltetes Netz

Eine Erweiterung des gefalteten Netzes. Die ersten beiden Faltungsschichten haben einen Kernel von Ausmaß vpm (5, 5) und einen Stride von (1, 1). Die letzte Faltungsschicht einen Kernel von (3, 3) und (1, 1).
Die erset Faltungsschicht verwendet hierbei 64 Kernel, die letzten beiden 32.
Alle Max-Pooling-Schichten haben ein Ausmaß und einen Stride von (2, 2).
Die Dropout-Wahrscheinlichkeit der Dropout-Schicht liegt bei 0,5.

3.5.4 Reduziertes gefaltetes Netz (reduced_cnn)

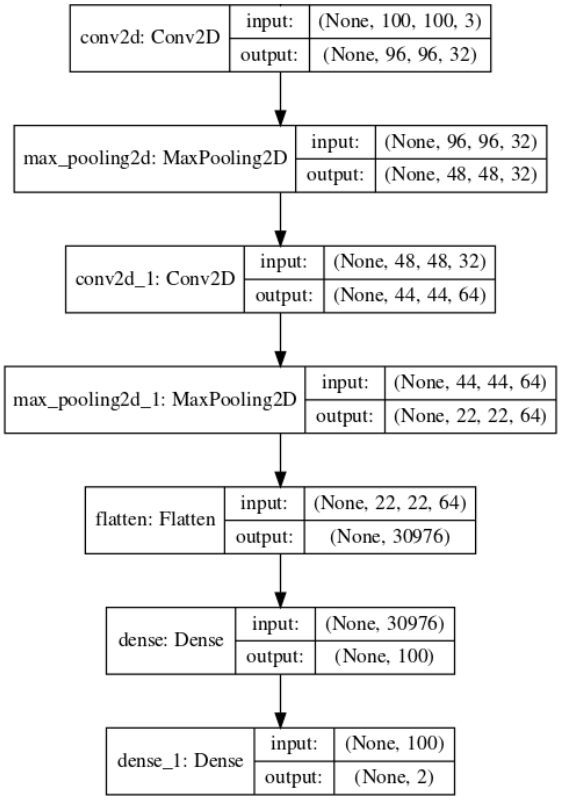


Abbildung 3.35: Netzmodell: Reduziertes gefaltetes Netz

Eine Reduzierung des Gefalteten Netzes zur Vermeidung von Overfitting.
Ausmaß und Stride der Kernel beider Faltungsschichten liegt bei (5, 5) und (1, 1).
Die erste Faltungsschicht verwendet 32 Kernel, die letzte 64. Ausmaß und Streide beider Max-Pool-Schichten liegt ebenfalls bei (2, 2) und (2, 2).

3.5.5 Reduziertes gefaltetes Netz 2 (reduced_cnn2)

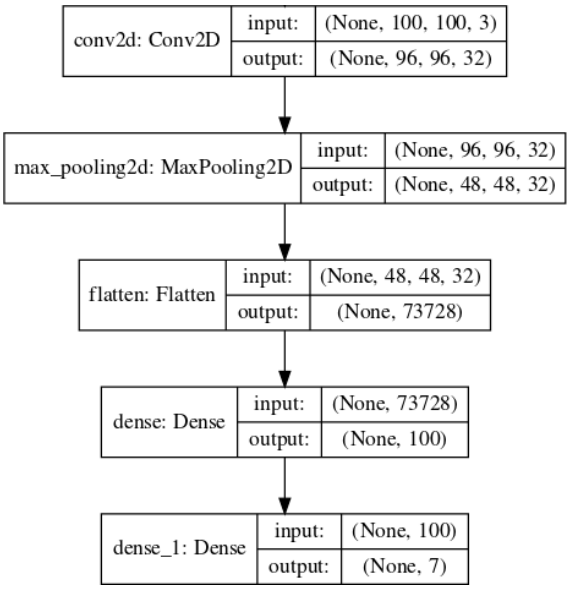


Abbildung 3.36: Netzmodell: Reduziertes gefaltetes Netz 2

Eine weitere Reduzierung des gefalteten Netzes.
32 Kernel mit den Ausmaßen und Stride von (5, 5) und (1, 1).
Die Max-Pool-Schicht (2, 2) und (2, 2).

3.5.6 Dichtes Netzwerk (dense)

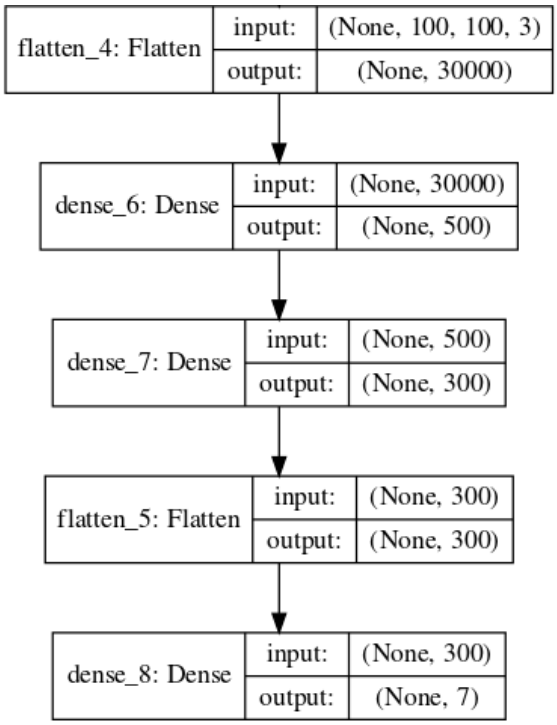


Abbildung 3.37: Netzmodell: Dichtes Netzwerk

Ein reines Perzeptron ohne Faltungsschichten.

3.5.7 Reduziertes dichtes Netzwerk (reduced_dense)

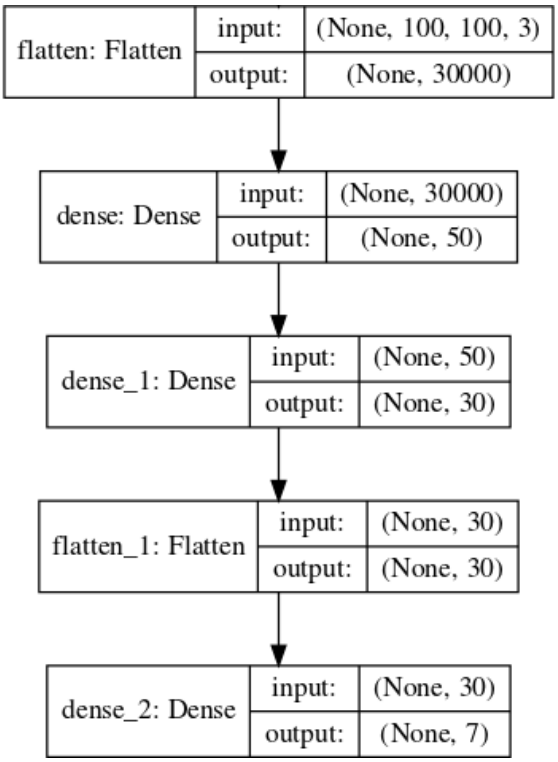


Abbildung 3.38: Netzmodell: Reduziertes dichtes Netzwerk

Eine Reduzierung des dichten Netzwerkes zur Vermeidung von Overfitting.

3.5.8 AlexNet (AlexNet)

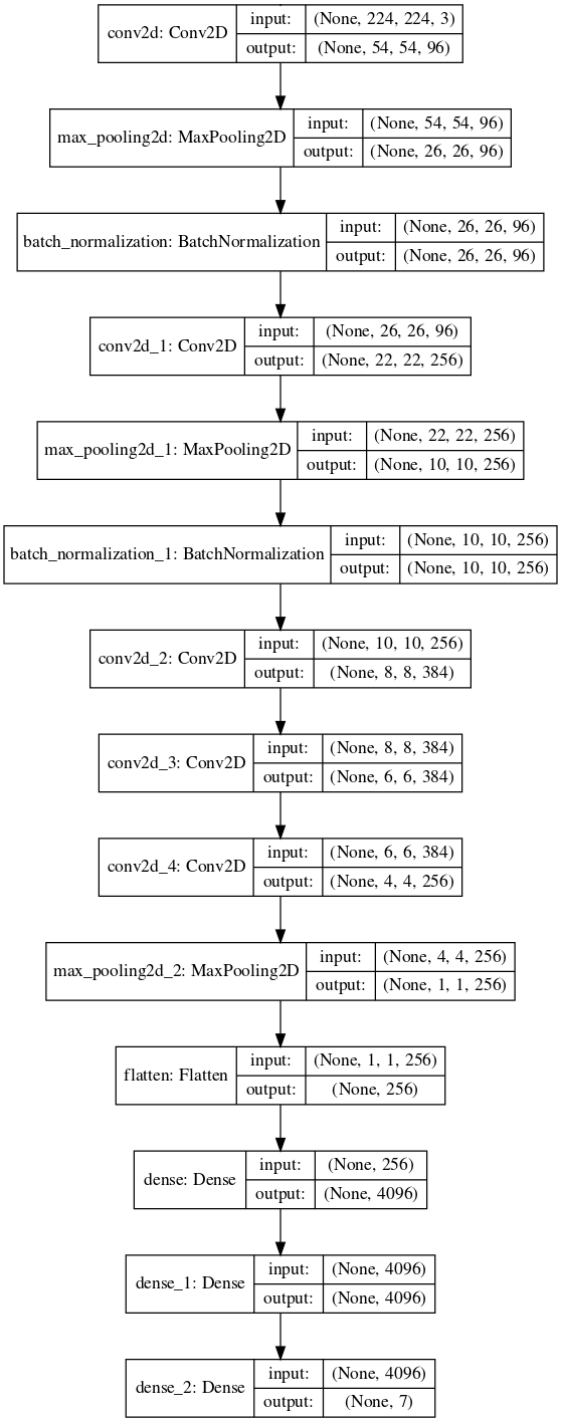


Abbildung 3.39: Netzmodell: AlexNet

Die Implementation des AlexNet. Alle Schichtparametrisierungen sind Originalgetreu übernommen worden.

3.6 Training

Die in dieser Arbeit vorgestellten Netze wurden ausschließlich mittels Backpropagation trainiert. Backpropagation stellt eine Form des überwachten Lernens für neuronale Netze dar, in der die errechneten Werte in den Ausgabeneuronen mit den Zielwerten verglichen werden, und mittels Ableitung der Aktivierungsfunktionen der Neuronen eine Verringerung des Fehlerwertes erreicht wird. Eine ausführliche Beschreibung dieser Lerntechnik wurde bereits 1986 im Nature Magazin gedruckt, und in Sektion 2.5 ausführlicher beschrieben [11].

Weiterhin wurden zum Training verschiedene Batch-Größen verwendet. Die Batch-Größe bestimmt die Anzahl an Trainingsdatensätzen, die durch ein neuronales Netz propagiert werden, bevor mittels Backpropagation eine Fehlerminimierung durchgeführt wird. Die Batch-Größe hat hierbei Einfluss auf die Speicherauslastung und Dauer des Trainings sowie auch die Leistung des Netzes. Folgende Werte wurden im neuronalem Netz cnn mit der Datenmange A mit unterschiedlicher Batch-Größe nach 50 Trainingsepochen auf einer GPU ermittelt.

	1	8	16	32
Trainingsdauer	1h 42m 31,14s	24m 30,38s	20m 35,54s	21m 40s
Genauigkeit (Korrekt/Gesamt)	866/2008	1151/2008	1114/2008	1127/2008

Tabelle 3.4: Vergleich: Batch-Größen

Aus den Ergebnissen in Tabelle 3.4 wird ersichtlich, dass eine gut gewählte Batch-Größe sowohl für die Trainingsdauer als auch für die Qualität des Netzes eine große Bedeutung hat. Das beste Ergebnis wurde mit einer Batch-Größe von 8 erzielt, mit zudem einer deutlich besseren Trainingszeit. Abgesehen von der allgemeinen Genauigkeit des Modells ist jedoch auch ein detaillierter Blick auf die Ergebnisse der einzelnen Klassen notwendig, um das Verhältnis von Batch-Größe, unausgeglichenen Klassen und allgemeiner Genauigkeit besser zu verstehen. Hierbei ist besonders hervorzuheben, dass das hier verwendete Modell bei einer Batch-Größe von 1 alle Bilder den Bogenbrücken zuordnete. Dieses Phänomen tritt jedoch nicht bei einer geringeren Anzahl Epochen auf.

Die Dauer des Trainings eines neuronales Netzes ist abhängig von der Anzahl der Neuronen, der Layer und der Datensätze, sowie die Architektur des Netzes und der Größe der Batches. Die Wahl der Hardware spielt aber ebenfalls eine Rolle. Moderne Grafikkarten trainieren bedeutend schneller als CPUs, wie die Ergebnisse in Tabelle 3.5 deutlich zeigt. Diese Ergebnisse wurden mit der Datenmenge A, 100 Epochen und einer Batch-Größe von 8 erzielt.

Trainingsdauer	CPU (i5-2500K)	GPU (GeForce GTX 680)
Batch	0,326s	0,035s
Epoch	842s	90s
Vollständiger Datensatz	4h 19m 16,31s	30m 57,22s

Tabelle 3.5: Vergleich: Performance CPU/GPU

Die Ergebnisse zeigen, dass das Training auf der GPU deutlich schneller verläuft, weswegen im weiteren Verlauf der Arbeit ausschließlich auf der GPU trainiert wurde.

Zum Training der Netzwerke wurden ein Python-Script entwickelt, dessen Quelltext sich in Anhang A befindet. Es kann ebenfalls auf folgendem Github-Repository gefunden werden:
https://github.com/jkrause1/BridgeTrainer

3.7 Qualitätskriterien

Zur Bestimmung der Klassifizierungsgüte eines neuronalen Netzes müssen quantifizierbare Merkmale definiert werden. Für diese Merkmale sollten weiterhin Vergleichswerte existieren, die eindeutig bestimmen, ob ein Netz besser klassifiziert als eine andere Methode, und somit tatsächlich eine Verbesserung darstellt.

Bei Klassifizierungsnetzwerken können drei Eigenschaften zur Quantifizierung der Klassifizierungsgüte herangezogen werden: Die *Präzision (precision)*, *Rückruf (recall)* und die *Genauigkeit (accuracy)*. Die Präzision und der Rückruf wird hierbei für jede Klasse einzeln berechnet, mit der ein Netzwerk ein Bild klassifizieren kann, während die Genauigkeit für alle Klassifizierungen insgesamt ausgerechnet wird. Alle drei Merkmale berechnen sich hierbei aus der Anzahl der TP (true positives), FP (false positives) und FN (false negatives). Bezogen auf eine Klasse A und ein Bild b ist ein TP ein Bild $b \in A$, dass korrekt der Klasse A, ein FP ein Bild $b \notin A$, dass fälschlich der Klasse A und ein FN ein Bild der Klasse $b \in A$ welches nicht der Klasse A zugeordnet wurde.

Präzision, Rückruf und Genauigkeit werden hierbei folgendermaßen berechnet wobei N für die Gesamtanzahl der Bilder steht:

$$precision_A = \frac{TP_A}{TP_A + FP_A}$$

$$recall_A = \frac{TP_A}{TP_A + FN_A}$$

$$accuracy = \frac{TP}{N}$$

Es lässt sich also sagen, dass die Präzision bestimmt, wieviele Zuweisungen zu Klasse A korrekt waren, der Rückruf, wieviele Bilder der Klasse A auch als solche erkannt wurden, und die Genauigkeit, wieviele Bilder über alle Klassen hinweg insgesamt korrekt klassifiziert wurden.

Diese Werte eines Netzwerkes werden mit der sogenannten *Baseline* verglichen. Die Baseline stellt Vergleichswerte für die Präzision, den Rückruf und die Genauigkeit zur Verfügung. Diese Werte werden ermittelt, in dem eine zufällige Klassifizierung aller Bilder angenommen wird. Bei sieben verschiedenen Klassen besteht hierbei eine Wahrscheinlichkeit von 14,29%, dass ein Bild zufällig korrekt klassifiziert wurde. Bei 2008 zu klassifizierenden Bildern (Siehe: Tabelle 3.2) errechnen sich hieraus folgende Werte:

	Menge	Zuweisunger	TP	FP	FN	Präzision	Rückruf
A	923	287	132	155	791	46%	14,3%
B	93	287	13	274	80	4,53%	14%
C	17	287	2	285	15	0,7%	11,76%
D	866	287	124	163	743	43%	14,31%
E	42	287	6	281	36	2,1%	14,29%
F	44	287	6	281	38	2,1%	13,64%
G	23	286	3	283	20	1,05%	13,04%

Tabelle 3.6: Tabelle: Baseline (Menge A-E)

und eine Genauigkeit von 14,24%.

Bei einem Klassifizierungsdurchgang mit zufälliger Verteilung konnten diese Werte bestätigt werden, wie in Abbildung 3.40 zu sehen ist.

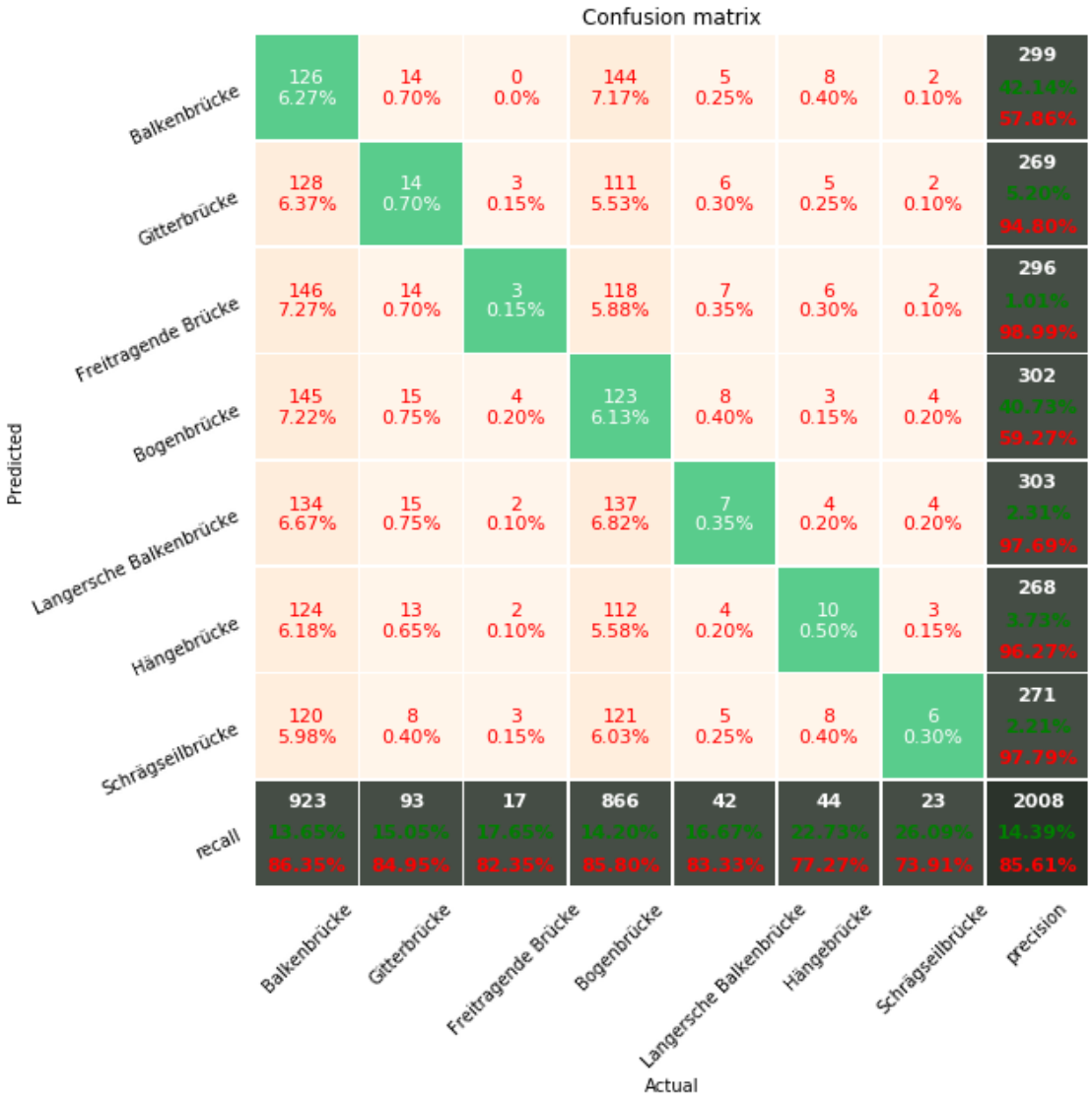


Abbildung 3.40: Konfusionsmatrix: Zufällige Verteilung (Mengen A-E)

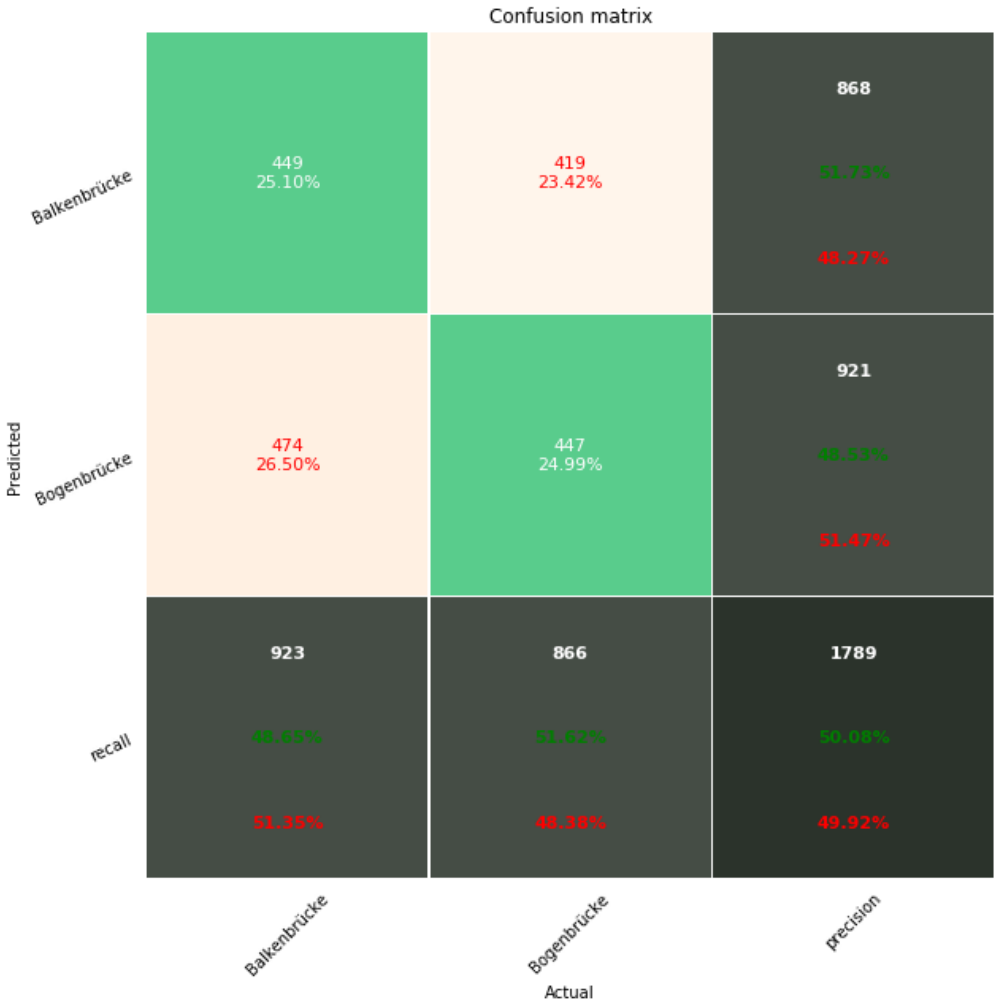


Abbildung 3.41: Konfusionsmatrix: Zufällige Verteilung (Menge F)

Für die Bildmenge F (Siehe Tabelle: 3.3) ergeben sich aufgrund der Tatsache, dass nur zwei Klassen existieren, folgende Werte:

	Menge	Zuweisunger	TP	FP	FN	Präzision	Rückruf
A	923	895	462	433	461	51,62%	50,05%
B	866	894	433	163	433	48,43%	50%

Tabelle 3.7: Tabelle: Baseline (Menge F)

und eine Genauigkeit von 50,02%.

Auch diese Werte konnten experimentell bestätigt werden, wie in Abbildung 3.41 zu sehen.

3.8 Ergebnisse

Die meisten Modelle zeigen unter den verwendeten Parametern eine Verbesserung der Genauigkeit im Vergleich zur Baseline. Diese sind grün gekennzeichnet. Eine Verschlechterung wird rot dargestellt. Diese Tabelle gibt keinen Eindruck in die Präzision und den Rückruf jeder einzelnen Klasse. Konfusionsmatritzen zu jedem einzelnem Versuch können auf der DVD unter Anhang A eingesehen werden.

- **Mod.** - Modell
- **Aufl.** - Bildauflösung
- **Ep.** - Anzahl Trainingsepochen
- **Ba.** - Batchgröße
- **Lr.** - Lernrate
- **Opt.** - Lernalgorithmus
- **Bm.** - Bildmenge

Mod.	Aufl.	Ep.	Ba.	Lr.	Opt.	Bm.	Genauigkeit
1l_dense	100x100	5	1	0,01	sgd	F	51,59%
1l_dense	100x100	5	8	0,01	adam	F	51,59%
1l_dense	100x100	5	8	0,01	rmsprop	F	51,59%
1l_dense	100x100	5	8	0,001	rmsprop	F	51,59%
1l_dense	100x100	5	8	0,01	sgd	F	51,59%
1l_dense	100x100	5	16	0,01	sgd	F	51,59%
1l_dense	100x100	20	8	0,01	sgd	F	51,59%
1l_dense	100x100	25	16	0,01	sgd	F	48,35%
1l_dense	100x100	100	1	0,01	sgd	F	48,35%
cnn	100x100	6	8	0,01	sgd	A	56,57%
cnn	100x100	30	1	0,01	sgd	A	53,24%
cnn	100x100	50	1	0,01	sgd	A	43,13%
cnn	100x100	50	8	0,01	sgd	A	57,32%
cnn	100x100	50	16	0,01	sgd	A	55,48%
cnn	100x100	50	32	0,01	sgd	A	56,13%
cnn	100x100	50	8	0,01	sgd	D	55,23%
cnn	100x100	8	1	0,01	sgd	E	16,83%
cnn	100x100	30	1	0,01	sgd	E	20,17%
cnn	100x100	30	30	0,01	sgd	E	23,36%
cnn	100x100	50	1	0,01	sgd	E	20,37%
cnn	100x100	50	8	0,01	sgd	E	18,08%
cnn	100x100	3	8	0,1	sgd	F	56,40%
cnn	100x100	3	8	0,0001	sgd	F	51,87%
cnn	100x100	5	8	0,1	rmsprop	F	48,41%
cnn	100x100	5	8	0,001	rmsprop	F	51,59%

cnn	100x100	5	8	0,1	sgd	F	58,58%
cnn	100x100	10	8	0,01	sgd	F	65,01%
cnn	100x100	10	8	0,01	sgd	F	53,49%
cnn	100x100	20	8	0,0001	sgd	F	57,97%
cnn	100x100	30	8	0,01	sgd	F	65,74%
cnn	100x100	50	8	0,001	rmsprop	F	48,41%
cnn_extended	100x100	50	8	0,0001	sgd	F	55,11%
cnn_extended	100x100	240	8	0,0001	sgd	F	63,78%
cnn_extended	100x100	500	8	0,0001	sgd	F	69,09%
cnn_extended	100x100	600	8	0,0001	sgd	F	71,16%
cnn_extended	100x100	2000	8	0,0001	sgd	F	70,71%
dense	100x100	10	8	0,01	sgd	F	54,79%
dense	100x100	20	8	0,01	sgd	F	53,88%
dense	100x100	50	8	0,001	rmsprop	F	51,59%
dense	100x100	50	16	0,001	adam	F	48,41%
reduced_dense	100x100	5	8	0,001	rmsprop	F	48,41%
reduced_dense	100x100	20	8	0,001	adam	F	51,59%
reduced_dense	100x100	20	8	0,01	sgd	F	54,72%
reduced_dense	100x100	20	16	0,001	rmsprop	F	51,93%
reduced_dense	100x100	50	16	0,001	rmsprop	F	48,41%
reduced_cnn	100x100	20	8	0,01	sgd	F	64,17%
reduced_cnn	100x100	30	8	0,01	sgd	F	62,49%
reduced_cnn2	100x100	20	8	0,01	sgd	F	59,42%
AlexNet	224x224	50	8	0,0001	sgd	A	63,4%
AlexNet	224x224	500	8	0,0001	sgd	A	54,98%
AlexNet	224x224	45	8	0,0001	sgd	B	41,73%
AlexNet	224x224	200	8	0,0001	sgd	B	52,99%
AlexNet	224x224	1000	8	0,0001	sgd	B	27,99%
AlexNet	224x224	1000	8	0,0001	sgd	C	61,25%
AlexNet	224x224	200	8	0,0001	sgd	D	57.92%
AlexNet	224x224	200	8	0,0001	sgd	E	25,10%
AlexNet	224x224	20	8	0,0001	sgd	F	65,79%
AlexNet	224x224	42	1	0,0001	sgd	F	71,83%
AlexNet	224x224	42	8	0,0001	sgd	F	73,09%
AlexNet	224x224	42	16	0,0001	sgd	F	67,58%
AlexNet	224x224	1000	8	0,0001	sgd	F	73,90%

Tabelle 3.8: Auflistung: Klassifizierungsergebnisse

Die besten Klassifizierungsergebnisse konnte das AlexNet mit allen Datenmengen erzielen. Hervorzuheben ist, dass die Ausbalancierung der Datenklassen mittels

Reduzierung über alle Modelle die schlechtesten Werte hervorrufte. Batch-Größen kleiner als 8 hatten ebenfalls einen detrimentalen Effekt. Ein positiver Effekt augmentierter Datenmengen zur Ausbalancierung von Datenklassen konnte nicht festgestellt werden. Alle Modelle zeigten entweder eine ähnliche oder sogar bessere Genauigkeit, wenn die unbalancierte Datenmenge verwendet wurde. Eine kleinere Lernrate führte ebenfalls durchgängig zu besseren Ergebnissen.

4 Schlussbetrachtungen und Ausblick

Diese Abschlussarbeit beschäftigte sich damit, einen Datensatz an Brückenbildern zu generieren, zu annotieren und von künstlichen neuronalen Netzen klassifizieren zu lassen. Hierfür konnte nicht auf einen bereits existierenden Datensatz zurückgegriffen werden. Für die Annotation der Bilde wurde eine eigene Software entwickelt, die ein verteiltes Arbeiten mehrere Annotierer auf einer Online-Plattform ermöglicht, da frei verfügbare Alternativen nicht den Anforderungen für diese Aufgabe genügten. Dem entstandenem Ungleichgewicht der Klassen wurde versucht, mit Bildaugmentierung zu begegnen, und abschließend mehrere Versuche mit selbst entworfenen und bekannten Modellen künstlicher neuronaler Netze vorgenommen, mit unterschiedlicher Parametrisierung und Datenmengen die besten Klassifizierungsergebnisse zu erzielen. Es wurde eine ausführliche Einführung in die Thematik der künstlichen neuronalen Netze in dieser Arbeit niedergeschrieben und die Ergebnisse veröffentlicht, verglichen und Schlußfolgerungen gezogen, welche Parameter das Training der Netze positiv beeinflusst. Damit hat diese Arbeit alle in der Aufgabenstellung definierten Ziele erreicht.

Obwohl in fast allen Versuchen im Vergleich zur Baseline eine insgesamt bessere Genauigkeit der Klassifizierungen erreicht wurde, existiert weiterhin viel Raum für Verbesserung, wenn künstliche neuronale Netze zur Brückenerkennung effektiv zur Abmilderungen von Auswirkungen von Katastrophenfällen beitragen sollen. Die Ergebnisse zeigen, dass die vorgenommenen Augmentationen zum Ausgleichen der Inbalance zwischen den Bilderklassen keine nennenswerten Effekte hatten, und die unaugmentiert am häufigsten vorkommenen Bilderklassen auch am häufigsten korrekt klassifiziert wurden. Die höchste Genauigkeit wurde daher auch erzielt, wenn ausschließlich mit Balken- und Bogenbrücken gearbeitet wurde. Die Anzahl

Eine mögliche Lösung dieses Problems besteht in der Erhöhung der Datenmenge. Von den 151.072 aus den Datenbanken von `http://www.geograph.org.uk` extrahierten Bilder wurden zum Training der in dieser Arbeit aufgelisteten Netzwerke nur 10.030 verwendet. Weiterhin können die Parameter der Bilderaugmentation angepasst werden, um diversere Bilder zu erzeugen.

Ein weiterer möglicher Aspekt der eine Revision erfordern könnte ist der Prozess der Annotation. Brückenbilder wurden von unterschiedlich qualifiziertem Personen ausschließlich nach Form und Struktur annotiert. Größe, Funktion der Brücke, Perspektive und Bildanteil waren keine definierten Kriterien. In ihrem 2015 erschienenem Paper zur Geschichte der ImageNet Large Scale Visual Recognition Challenge unterstrichen die Autoren die Wichtigkeit korrekt annotierter Datensätze für die korrekte Klassifizierung:

> „The key lesson of collecting the datasets and running the challenges for five years is this: **All human intelligence tasks need to be exceptionally well designed.** We learned this lesson both when annotating the dataset using Amazon Mechanical Turk workers (Section 3) and even when trying to evaluate human level image classification accuracy using expert labelers (Section 6.4). The first iteration of the labeling interface was always bad – generally meaning *completely unusable*. If there was any inherent ambiguity in the questions posed (and there almost always was), workers found it and accuracy suffered. If there is one piece of advice we can offer to future research, it is to very carefully design, continuously monitor, and extensively sanity-check all crowdsourcing tasks.“ (Russakovsky 2015:33)

Die bisherigen Ergebnisse lassen jedoch darauf hoffen, dass mit weiterführender Forschungsarbeit akzeptable Ergebnisse für alle Brückenkategorien erreicht werden können. Diese Bachelorarbeit hat hierfür umfangreiche Vorarbeit in Form neu entwickelter Software geleistet, die zur Annotation von Bilderdaten und dem Training von Netzwerken eingesetzt werden kann. Bestandteil zukünftiger Arbeiten im Rahmen des Gesamtprojektes könnten sein:

- Verbesserung der Genauigkeit, Rückruf und Präzision aller Klassen,
- Implementierung der neuronalen Netze auf ein Smartphone und Klassifizierung verschiedener Brücken live vor Ort,
- Ergänzung des generierten Datensatzes um weitere Brückenbilder,
- Ermitteln optimaler Kriterien zur Beschaffenheit von Brückenbilder zur Annotation,
- Finden neuer und besserer Strategien zur Abmilderung negativer Effekte des Klassen-Ungleichgewichtes oder Ergänzung der verwendeten Augmentationen.

Abschließend soll erwähnt werden, dass alle in dieser Arbeit entwickelte Software Anhang A angefügt und für zukünftige Arbeiten verwendet und weiterentwickelt werden können und dürfen. Der generierte Datensatz wird zur Unterstützung anderer Forschungsarbeiten nach Abgabe dieser Arbeit auf der Online-Plattform kaggle veröffentlicht. Von dort kann er zur Reproduzierung und Weiterverarbeitung bezogen werden.

Anhang A: DVD

Die der Bachelorarbeit beiliegende DVD beinhaltet Folgendes:

- *Johannes_Krause_BA_2019.pdf* – Eine digitale Kopie dieser Bachelorarbeit
- *BeamBridgeInn* – Quellcode der Annotierungssoftware „Beam-Bridge-Inn“
- *BridgeAugmentor* – Quellcode des Python-Scripts zur Augmentierung von Bildern
- *BridgeTrainer* – Quellcode des Python-Scripts zum Training von neuronalen Netzen und Bilderklassifizierung
- *DatasetBalancer* – Quellcode des Python-Scripts dass zur Ausbalancierung der Datenmengen verwendet wurde
- *NeuralNets* – Eine gepackte Ordnerstruktur die die Modelle, trainierte Gewichte und Ergebnisse aller vorgestellten Modelle enthält

Literaturverzeichnis

[1] Albelwi, Saleh ; Mahmood, Ausif: A Framework for Designing the Architectures of Deep Convolutional Neural Networks. In: entropy 19 (2017), Nr. 6

[2] Angelica, Amara D. ; Schmidhuber, Jürgen: How bio-inspired deep learning keeps winning competitions. URL: <http://www.kurzweilai.net/how-bio-inspired-deep-learning-keeps-winning-competitions>, 18.09.2018

[3] Bloice, Marcus D. ; Stocker, Christof ; Holzinger, Andreas: Augmentor: An Image Augmentation Library for Machine Learning, arXiv preprint arXiv:1708.04680, URL: <https://arxiv.org/abs/1708.04680>, 23.01.2019

[4] Caruana, Rich ; Lawrence, Steve ; Giles, Lee: Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping. URL: <http://papers.nips.cc/paper/1895-overfitting-in-neural-nets-backpropagation-conjugate-gradient-and-early-stopping.pdf>, 28.09.2018

[5] Dahl, Ryan <rld@google.com> ; Norouzi, Mohammad <mnorouzi@google.com> ; Shlens, Jonathon <shlens@google.com> : Pixel Recursive Super Resolution. URL: <https://arxiv.org/abs/1702.00783v2>, 13.09.2018

[6] LeCun, Yann ; Bottou, Leon ; Orr, Genevieve B. ; Müller, Klaus-Robert: Efficient backprop. In: Neural Networks: Tricks of the trade (2012), S. 9-48

[7] Benndorf, Maik ; Garsch, Maximillian ; Haenselmann, Thomas ; Gebbeken, Norbert ; Videkhina, Inna : Mobile bridge integrity assesment. URL: <https://doi.org/10.1109/ICSENS.2016.7808728>, 13.09.2018

[8] Benndorf, Maik ; Haenselmann, Thomas ; Garsch, Maximillian ; Gebbeken, Norbert ; Mueller, Christian A. ; Fromm, Tobias ; Luczynski, Tomasz ; Birk, Andreas: Robotic bridge statics assessment within strategic flood evacuation planning using low-cost sensors. URL: <https://doi.org/10.1109/SSRR.2017.8088133>, 13.09.2018

[9] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: The bulletin of mathematical biophysis 4 (1943), Nr. 5, S. 115-133

[10] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks In: Advances in Neural Information Processing Systems 25 (2012), S. 1097-1105 URL:

<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

[11] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by backpropagating errors. In: Nature (1986), Nr. 323, S. 533-536

[12] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li : ImageNet Large Scale Visual Recognition Challenge (2015). URL: <https://arxiv.org/abs/1409.0575v3>, 20.09.2018

[13] Schmidhuber, Jürgen : History of computer vision contests won by deep CNNs on GPU. URL: <http://people.idsia.ch/ juergen/computer-vision-contests-won-by-gpu-cnns.html>, 13.09.2019

[14] Ioffe, Sergey ; Szegedy, Christian : Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. URL: <https://arxiv.org/pdf/1502.03167>, 23.09.2018

[15] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Droput: A Simple Way to Prevent Neural Networks from Overfitting. In: Journal of Machine Learning Research (2014), Nr. 15, S. 1929-1958

[16] Suwajanakorn, Supasorn ; Seitz, Steven M. ; Kemelmacher-Shlizerman, Ira: Synthesizing Obama: Learning Lip Sync from Audio. In: ACM Transactions on Graphics 36 (2017), Nr. 4, Article 95 - ISSN 0730-0301

[17] Bridge, Wikipedia URL: <https://en.wikipedia.org/wiki/Bridge#Structure_type>, 13.08.2018

[18] Zeiler, Matthew D. <Zeiler@cs.nyu.edu> ; Fergus, Rob <Fergus@cs.nyu.edu> : Visualizing and Understanding Convulational Networks. URL: <https://arxiv.org/pdf/1311.2901v3.pdf>, 13.09.2018

[19] Zhang, Chiyuan <chiyuan@mit.edu> ; Bengio, Samy <bengio@google.com> ; Hardt, Morizt <mrtz.google.com> ; Recht, Benjamin <brecht@berkeley.edu> ; Vinyais, Oriol <vinyals@google.com> : Understanding deep learning requires rethinking generalization. URL: <https://arxiv.org/abs/1611.03530>, 13.09.2018

[20] R. Kidd: A50 bridge across the trent and mersey canal, derbyshire. URL: <http://www.geograph.org.uk/photo/1616022.>, 09.09.2018

[21] Halling, Philip: Bridge 238, Oxford Canal. URL: <http://www.geograph.ie/photo/1388675>, 09.09.2018

[22] Anstiss, David: Queen Elizabeth Bridge goes over River Thames. URL: <http://www.geograph.ie/photo/1169383>, 09.09.2018

[23] Chadwick, N.: Railway Bridge over the River Medway (2). URL: <http://www.geograph.ie/photo/1159042>, 09.09.2018

[24] Bone, Robert: Railway Bridge over the Shannon, Athlone. URL: <http://www.geograph.ie/photo/113118>, 09.09.2018

[25] Roberts: Fred. Horkstow Bridge. URL: <http://www.geograph.ie/photo/10846>, 09.09.2018

[26] Nicholls, Bill: Wallingford bridge. URL: <http://www.geograph.ie/photo/1059473>, 09.09.2018

[27] Armitstead, John: Bridge at Dundrum. URL: <http://www.geograph.org.uk/photo/125237>, 09.09.2018

[28] Williams, Phil: Weak Bridge. URL: <http://www.geograph.org.uk/photo/110368>, 20.09.2018

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 28.01.2019

Johannes Krause