# Inhaltverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ADAM | Adaptive moment estimation |
| CNN | Convolutional neural network |
| KI | Künstliche Intelligenz |
| MLP | Multi layer perceptron |
| SGD | Stochastic gradient descent |

# 1 Einleitung und Aufgabenstellung

Mit rund 100 Milliarden Neuronen, die mit 1x10^12 Synapsen in unterschiedlicher Dichte miteinander verschaltet sind, ist das menschliche Gehirn das komplexeste Organ, das die Natur hervorgebracht hat. Zahlreiche Prozesse wie beispielsweise die Muster- und Bilderkennung, die Spracherkennung und das Gedächtnis, können innerhalb kürzester Zeit abgerufen und in Folge kontinuierlich trainiert werden. Die Struktur, die dies möglich macht wird als neuronales Netz bezeichnet und steht sowohl für die Industrie als auch für die Forschung seit Jahrzehnten im Fokus. Mit Sicht auf die industrielle Entwicklung hin zu Industrie 4.0 und die damit verbundenen komplexer werdenden Aufgaben und Produktionsstrategien fordern mehr autonome Entscheidungen, schnellere Erkennung und darüber hinaus auch das Lernen aus bereits durchgeführten Prozessen. Im Zuge der Erforschung der Strukturen von Neuronen und Synapsen, ist es Forschern bereits Mitte der 90 Jahre gelungen ein künstliches neuronales Netz (Backpropagation trainiertes vorwärts gerichtetes Netz) so zu trainieren, dass geschriebene Worte erkannt und sprachlich ausgegeben werden konnten (NETtalk). Heute finden künstliche neuronale Netze ihre Anwendung in den unterschiedlichsten Gebieten, überwiegend aber in dem Bereich der Bild- und Mustererkennung, sowie bei der Erstellung von Prognosen in komplexen Systemen beispielsweise der medizinischen Diagnostik.

Der steigende Bedarf an künstlichen Intelligenzen (KI) und künstlichen neuronalen Netzen (NN) hat nicht nur in der Industrie, Änderungen nach sich gezogen, auch die Ausrichtung der heutigen Ingenieurs-Studiengänge passt sich diesem Bedarf an. Für Lern-, Trainings- und Testzwecke hat das kanadische Institut “Canadian Institutes For Advanced Research” den heute meist verwendeten Datensatz (CIFAR-10 ) für maschinelles Lernen entwickelt und dient mit seinen 60.000 Bildern und 10 Klassen (6.000 Bilder pro Klasse) als Grundlage für die nachfolgenden Ausarbeitung. Die farbigen Bilder in der Größe von 32x32 Pixel können durch die vergleichbar geringe Größe, problemlos für die Untersuchung verschiedener künstliche neuronaler Netze und deren Parameter verwendet werden. Die verwendeten Netze sind aus bereits bestehenden Codes zusammengefügt und mit eigenen geschriebenen Bausteinen erweitert, um eine vergleichbare Ausgangssituation zu schaffen. Wie bereits erwähnt liegt das Hauptaugenmerk, der nachfolgenden Ausarbeitung, auf der Auswirkung der einzelnen Parameter bezüglich der Ergebnisse der Lern- bzw Fehlerrate.

# 2 Bewertungskriterien für die neuronalen Netze

In der Aufgabenstellung war es vorgeschrieben ein MLP, ein CNN und ein Netz mit Transfer-Learning zu programmieren. Da diese Netze alle auf denselben Datensatz programmiert worden sind, wurde zunächst überlegt, wie die verschiedenen Netze miteinander vergleichbar sind und auf welche Parameter dabei zu achten ist.

Zunächst sind für alle Netze die Accuracy bzw. die Genauigkeit grundlegend wichtig. Diese lässt sich auch bei allen Netzen gleich bewerten. Des Weiteren ist ein Bewertungskriterium die Epochenzahl und die benötigte Zeit zum Lernen. Diese beiden Punkte liegen nahe beieinander, jedoch ist z.B. eine geringe Anzahl an Epoche nicht immer ein Indikator für eine geringe Trainingszeit. Somit müssen diese beiden Parameter zusammen kritisch bewertet werden.

Ein weiterer Output aller Programm-Codes ist die Parameteranzahl über die gelernt wurde. Hieraus ist erkennbar, wie rechenaufwendig das Netz ist, wie viele Vernetzungen das Netz hat und somit auch wie komplex das Netz ist. Auch hier kann eine Bewertung in Verbindung mit der benötigten Zeit stattfinden.

Zum Schluss erhält man bei jedem Netz eine sog. Confusion-Matrix. Diese gibt Aufschluss darüber, in welcher Klasse die gelabelten Daten erkannt worden sind. Aus dieser Matrix lassen sich z.B. Fehler des Netzes ableiten.

# 3 Implementierte Programmabschnitte

In diesem Kapitel werden die von der Gruppe implementierten Programmcodes vorgestellt.

## 3.1 Early Stop

Um ein Overfitting beim Trainieren der neuronalen Netze zu vermeiden, wurde die Regulierungsmethode Early Stopping eingesetzt. Beim Early Stopping wird das Trainieren des neuronalen Netzes abgebrochen, wenn über mehrere Epochen keine Verbesserung der Accuracy mit den Validierungsdaten feststellbar ist.

Zur Umsetzung des Early Stoppings wird der Lernprozess mit einer while-Schleife ausgeführt, innerhalb der das neuronale Netz mit der Methode train_val(), aus der Vorlesung, eingelernt wird. Diese Methode gibt den Tensor val_corrects_history zurück, in dem der Verlauf der Accuracy mit den Validierungsdaten gespeichert wird. Nach jeder Epoche wird die Differenz aus der aktuellen Accuracy und der aus der vorherigen Epoche gebildet. Unterschreitet die Differenz der Accuracy-Werte mehrfach hintereinander einen Schwellwert, wird der Lernprozess abgebrochen. Da sich die Accuracy des Modells vor dem Abbruch des Trainings auch verschlechtern kann, darf als beste Version des neuronalen Netzes nicht das Modell der letzten Epoche verwendet werden. Innerhalb der while-Schleife wird nach jeder Epoche überprüft, ob das aktuelle Modell die größte Accuracy im Tensor val_corrects_history besitzt. Ist dies der Fall, wird der aktuelle Zustand des neuronalen Netzes unter der Variable best_model gespeichert.

Der Programmcode für das Early Stopping befindet sich im Anhang A: Überschrift.

## 3.2 Confusion-Matrix

Eine Confusion Matrix zeigt auf, welche Klassen von einem neuronalen Netz richtig erkannt werden und welche fehlerhaft eingestuft werden. Erstellt wird die Confusion Matrix mit den Validierungs- oder mit Testdaten. Eine Stichprobe wird bei einer Confusion Matrix immer in die Zeile, die dem Label entspricht und in die Spalte, die mit der Vorhersage übereinstimmt, eingetragen.

Das im Rahmen der Projektarbeit implementierte Programm zur Erstellung der Confusion Matrix klassifiziert die Validierungsdaten batchweise. Die Batches werden dem neuronalen Netz in mehreren Iterationsschritten durch Schleifen zugeführt. Die Klassifikation der Validierungsdaten erfolgt mit dem Befehl model(). Dieser gibt einen Tensor zurück, in dem die Wahrscheinlichkeiten für jede Klasse enthalten sind. Die Position des höchsten Wahr-

scheinlichkeitswerts im Tensor entspricht der Klasse des Eingangsbildes. Nach der Klassifikation eines Bildes wird in der Confusion Matrix der Wert an der Position [Label|Vorhersage] um 1 erhöht. Der Plot wird mit dem Befehl matshow() erzeugt.

Das Pythonskript zur Erstellung der Confusion Matrix ist im Anhang B: Überschrift enthalten.

## 3.3 Geänderte Parameter an den Netzen

Die Implementierung der neuronalen Netze erfolgte auf Basis der Vorlesungsbeispiele. Erstellt wurden fünf MLPs, fünf CNNs und fünf weitere neuronale Netze durch Transfer Learning. Die neuronalen Netze unterscheiden sich in den folgenden Parametern:

- Lernrate: Die Lernrate beeinflusst, wie stark die Gewichte und Schwellwerte nach dem Durchlaufen eines Trainingsdatenpaares oder eines Batches angepasst werden. Je größer die Lernrate ist, desto stärker werden die Parameter korrigiert. Wird die Lernrate zu groß gewählt, wird das Minimum der Lossfunction ggfs. nicht erreicht. Eine zu niedrige Lernrate verlangsamt den Trainingsprozess [1].
- Batchsize: Die Batchsize entspricht der Anzahl an Trainingsdatenpaaren, die das neuronale Netz beim Trainingsprozess durchlaufen, bevor die Gewichte und Schwellen korrigiert werden [1]. Durch das Trainieren des Modells mit Batches kann der benötigte Speicher reduziert werden und der Lernprozess erfolgt schneller [2].
- Optimizer: Ein Optimizer dient zur Berechnung der neuen Gewichte und Schwellen anhand der Lossfunction. Zum Einsatz kamen die Optimizer SGD und ADAM. Der SGD-Optimizer berücksichtigt bei der Korrektur eines Gewichts oder einer Schwelle die Gradienten der vorherigen Durchläufe. Hierdurch wird der Verlauf der Gewichts- und der Schwellwertanpassungen geglättet und das Minimum der Lossfunction wird schneller erreicht. Der ADAM-Optimizer baut auf dem SGD-Optimizer auf. Dieser verwendet zur Korrektur der Gewichte und Schwellen zusätzlich eine anpassbare Lernrate. Diese wird umso kleiner, je größer die Gradienten sind [1].
- Dropout: Dropout ist eine Regulierungsmethode. Durch den Einsatz von Dropout wird eine zufällige Auswahl an Neuronen beim Lernprozess deaktiviert. Hierdurch wird die Prozesszeit zum Lernen verkürzt und eine Überanpassung der Neuronen an die Trainingsdaten wird verhindert [1].
- Neuronenzahl, Schichtzahl: Weiterhin wurden die Anzahl an Neuronen sowie Schichten variiert. Es sollte untersucht werden, wie sich diese Parameter auf die Anpassungsfähigkeit des neuronalen Netzes auswirken.

# 4 Multi Layer Perceptron

## 4.1 Einführung zu MLPs

Ein Multi Layer Perceptron ergibt sich durch Zusammenschaltung von Neuronen, die in Schichten mit gerichtetem Informationsfluss angeordnet sind. MLP lassen sich hinsichtlich ihrer Topologie in zwei Klassen unterteilen:

**Feedforward Netze:** Eine Ausbreitung über das Netzwerk setzt sich nur in einer Richtung fort.

**Rekurrente Netze:** Eine Ausbreitung der Aktivierung in einer Schicht hat Auswirkungen auf die Aktivierung einer Früheren Schicht [1].

## 4.2 Aufbau des Basisnetzes

In der Abbildung 4-1 ist die MLP-Netz-Architektur dargestellt. Bei einem RGB-Bild (3 Kanäle) mit den Maßen 32∗32 Pixel ist die Neuronenzahl in der Input-Schicht durch die Größe vom Bild vorgegeben 32∗32∗3 = 3072. Die Neuronen benutzen für die Berechnung der Netzeingabe immer eine lineare Funktion. Als Aktivierungsfunktion in den Hidden Layers wird eine ReLu-Funktion verwendet. In den Hidden-Layers wird ein Dropout verwendet (im Basisnetzauskommentiert) um Overfitting zu vermeiden. Als Aktivierungsfunktion in der Ausgabeschicht wird eine Softmax-Funktion verwendet.

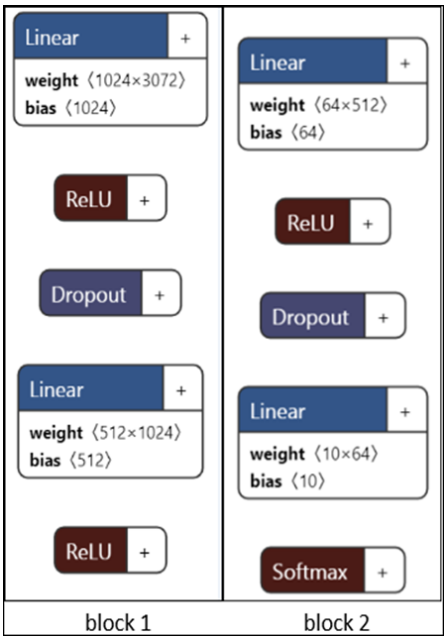


Abbildung 4-1: Architektur Basis-MLP

## 4.3 Anpassung der Hyperparameter und Ergebnisse

Die nachfolgende Tabelle (siehe Tabelle 4-1) gibt eine Übersicht über die verwendeten Paramater in den unterschiedlichen MLP-Netzen. Dabei wird von dem Basisnetz ausgegangen. Die zugehörigen Ergebnisse werden nachfolgend genauer erläutert.

Tabelle 4-1: Übersicht der Hyperparameter der MLP-Netze

| | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **Lernrate** | 0,0001 | 0,001 | 0,001 | 0,0001 | 0,00125 |
| **Batchgröße** | 128 | 128 | 128 | 128 | 128 |
| **Optimizer** | ADAM | ADAM | SGD | ADAM | ADAM |
| **Dropout** | Nein | 0,2 | Nein | Nein | Nein |
| **Neuronenzahl** | 1610 | 2210 | 1610 | 4810 | 1610 |
| **Schichtzahl** | 3 | 3 | 3 | 9 | 3 |

Das Basisnetz entspricht dem Netz aus der ersten Spalte.

**Lernrate:** Bei der Lernrate ist darauf zu achten, dass diese nicht zu groß oder zu klein gewählt wird. Bei einer zu groß gewählten Lernrate kann das Netz nicht lernen und liefert lediglich eine 10-prozentige Genauigkeit. Wird die Lernrate jedoch zu klein initialisiert, dauert das Lernen bedeutend länger und bleibt vermutlich bei lokalen Minima hängen. Die Abbildung 4-2 zeigt den Verlauf für die Accuracy bei der besten Lernrate aus der Versuchsreihe.

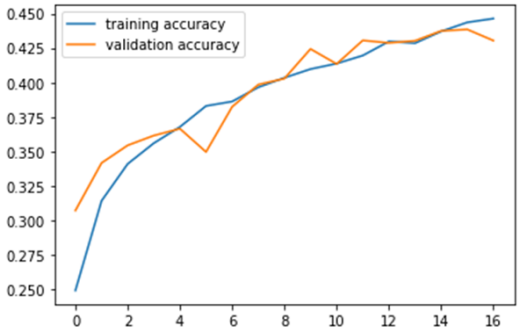


Abbildung 4-2: Verlauf der Accuracy bei Änderung der Lernrate

**Batchsize:** Bei einem Batch werden mehrere Daten / Bilder zu einem Trainingspaar zusammengefasst. Das bedeutet, dass nicht jedes Bild einzeln vom Netz gelernt wird, sondern mehrere Bilder zu einem Batch zusammengefasst werden und diese dann durch das Netz geleitet werden.

Versuche haben gezeigt, dass bei dem MLP-Netz eine Batchsize von 128 optimal ist und eine Genauigkeit von 47% liefert. (Abbildung 4-3)

Wird die Batch Size zu klein bzw. zu groß gewählt, werden die Ergebnisse ungenauer und bei viel zu kleiner oder viel zu großer Batchsize in Kombination mit einer zu kleinen/großen Learning Rate erkennt das Netz gar keine Klassen mehr.

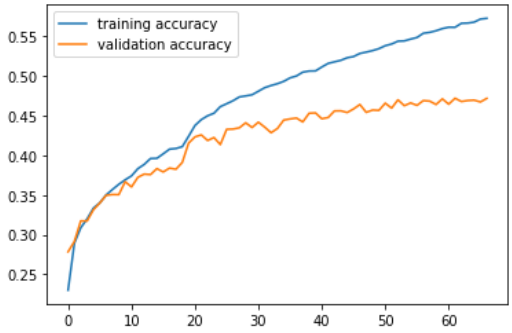


Abbildung 4-3: Verlauf der Accuracy bei Änderung der Batchsize

**Optimizer:** Das neuronale Netz mit dem Verlauf der Accuracy für die Lern- und Validierungsdaten aus der Abbildung 4-4 verwendet den Optimizer SGD mit Momentum. Der Einsatz des Optimizers führte anfangs zu einer sehr großen Steigung der Accuracy, sodass nach weniger als zehn Epochen bereits eine Genauigkeit von über 40% erreicht werden konnte. Andere Modelle benötigten hierfür die doppelte Anzahl an Epochen. Durch den Einsatz des SGD-Optimizers konnte dieses Netz die beste Accuracy mit den Validierungsdaten erzielen.

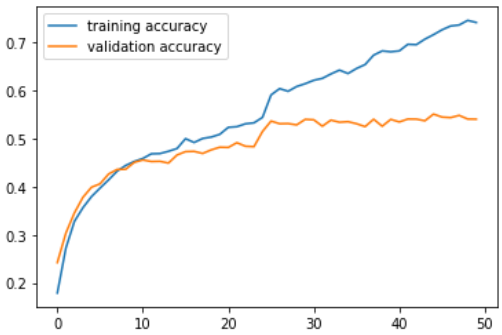


Abbildung 4-4: Verlauf der Accuracy bei Änderung des Optimizers

**Dropout:** Dropout stellt einen sehr effektiven Weg zum Training von tiefen neuronalen Netzen dar, dadurch kann die Gefahr von Overfitting verringert werden. Bei diesen neuronalen

Netze Versuche haben gezeigt, dass Dropout besseres Ergebnis beim CNN-Netz gegeben hat, seine Auswirkungen wurden beim MLP-Netz kaum erkannt.

**Neuronenzahl:** Bei der Parameteranpassung der Neuronenanzahl waren im Vergleich sehr stark Auswirkungen auf die Accuracy mit den Trainings- und mit den Validierungsdaten zu erkennen. Umso mehr Neuronen die erste schichten aufweist, desto höher sind die Accuracy-Werte in den ersten Epochen zu beobachten. Das beste Ergebnis konnte bei dieser Parameteranpassung erzielt werden, indem der erste Layer mit 1000, der zweite Layer mit 80 und der dritte Layer mit 20 Neuronen bestückt wurde (siehe Abbildung 4-5). Um zusätzlich die Auswirkungen der Drop Out Funktion zu erfahren, wurde jeder Trainingslauf einmal mit einem Dropout von 20% und einmal ohne durchgeführt. Abschließend wurde sichtbar, dass bei der Neuronennpassung keine positive Veränderung durch die Drop Out Funktion hervorgerufen wurde.

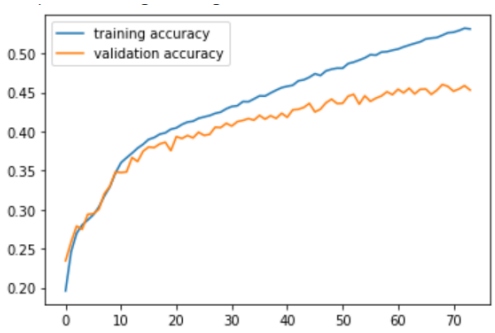


Abbildung 4-5: Verlauf der Accuracy bei Änderung der Neuronenzahl

**Schichtzahl:** Das tiefste programmierte neuronale Netz besitzt neun versteckte Schichten. Im Gegensatz zu den anderen MLPs besitzt dieses jeweils drei Schichten mit 1024, mit 512 und mit 64 Neuronen. Bei diesem neuronalen Netz beträgt die Parameteranzahl fast das doppelte der anderen. Verglichen zu den weiteren Modellen erzielt dieses eine bessere Accuracy mit den Trainings- und mit den Validierungsdaten, allerdings dauert der Lernprozess durch die große Menge an Parametern deutlich länger. Beim Trainingsprozess schwingt die Accuracy der Trainings- und der Validierungsdaten sehr stark. Dies erschwert den Eingriff durch das Early Stopping. Die Abbildung 4-6 zeigt der Verlauf der Accuracy der Lern- und der Validierungsdaten.

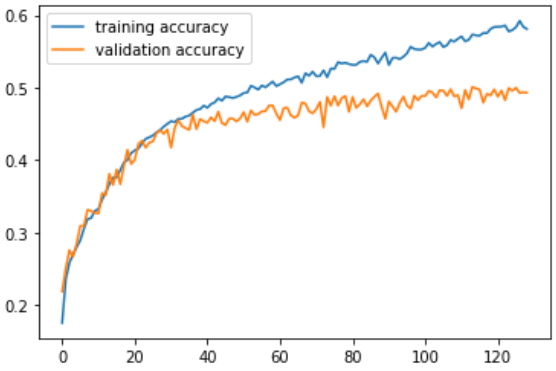


Abbildung 4-6: Verlauf der Accuracy bei Änderung der Schichtzahl

Nachfolgend (siehe Tabelle 4-2) sind die zuvor bestimmten Bewertungskriterien gegenübergestellt. Es ist zu erkennen, dass es zu starken Schwankungen der Laufzeit gekommen ist. Dabei ist besonders auf das Verhältnis von Epochenzahl und Laufzeit zu beachten, teilweise wurde über eine ähnliche Epochenzahl iteriert bei sehr unterschiedlicher Laufzeit. Selbst bei gleicher Parameterzahl werden unterschiedliche Laufzeiten benötigt.

Tabelle 4-2: Ergebnisse der MLPs in Abhängigkeit der Hyperparameter

| | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| **Epochenzahl** | 68 | 74 | 128 | 130 | 18 |
| **Laufzeit [s]** | 484 | 505 | 326 | 1090 | 203 |
| **Accuracy Trainingsdaten [%]** | 57,3 | 52,1 | 74,06 | 59,2 | 44,4 |
| **Accuracy Validierungsdaten [%]** | 47,2 | 45,3 | 55,04 | 50,1 | 43,9 |
| **Parameteranzahl (·10³)** | 3705 | 3154 | 3705 | 6338 | 3705 |

Die nachfolgende Confusion-Matrix (Abbildung 4-7) wurde aus dem MLP-Netz mit der besten Accuracy erzeugt. In der Matrix ist zu erkennen, dass z.B. Autos, Pferde und Schiffe recht gut erkannt wurden und im Gegenzug dazu, Katzen wiederum relativ schlecht. Außerdem ist zu erkennen, dass Trucks oft als Autos erkannt wurden. Dies kann auf die Ähnlichkeit an Merkmalen (z.B. Reifen) zurückzuführen sein.

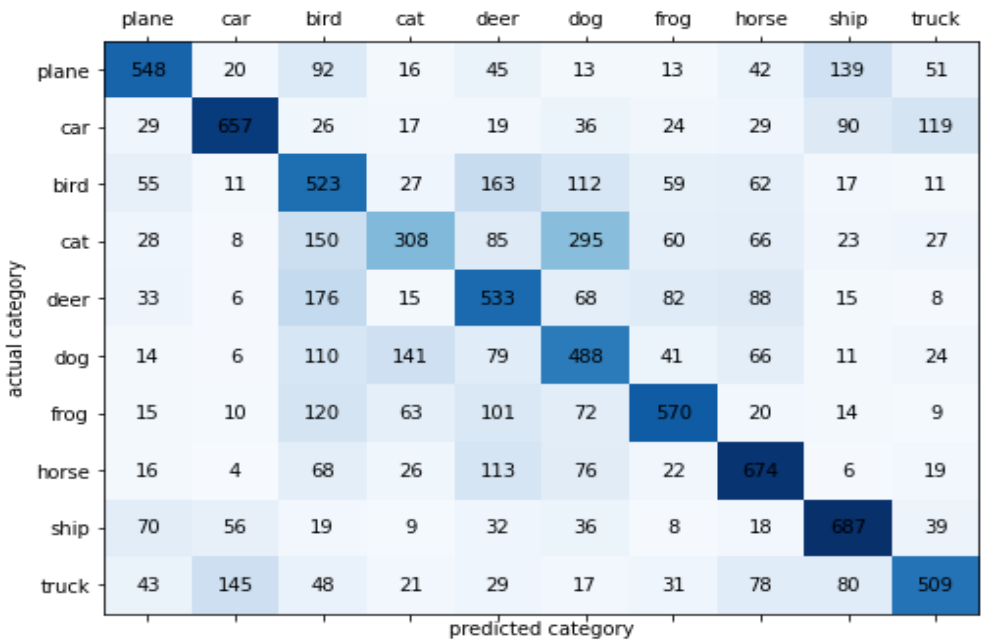


Abbildung 4-7: Confusion-Matrix für das beste MLP mit dem SGD-Optimizer

# 5 Convolutional Neuronal Network

## 5.1 Einführung zu CNNs und Aufbau des Basisnetzes

CNNs sind neuronale Netze für die Bilderkennung. Es handelt sich um Feed-Forward-Netze, die durch Backpropagation trainiert werden. Diese sind so konzipiert, dass visuelle Muster mit Hilfe von Filtern direkt aus Pixelbildern mit minimaler Vorverarbeitung erkannt werden [3].

In der Abbildung 5-1 ist das Basisnetz zu sehen. Dieses besitzt eine tiefe CNN-Netz-Architektur mit drei verschiedenen Blöcken an Konvolutionsschichten und der ReLU-Funktion als Aktivierung. Weiterhin wird in den Hidden Layers das MaxPooling verwendet. In der Ausgabeschicht wird statt der Softmax-Funktion eine lineare Funktion verwendet, da mit dieser bessere Ergebnisse erzielt wurden. Um die Gefahr von Overfitting zu verringern, wird Dropout und Batch Normalization eigesetzt.

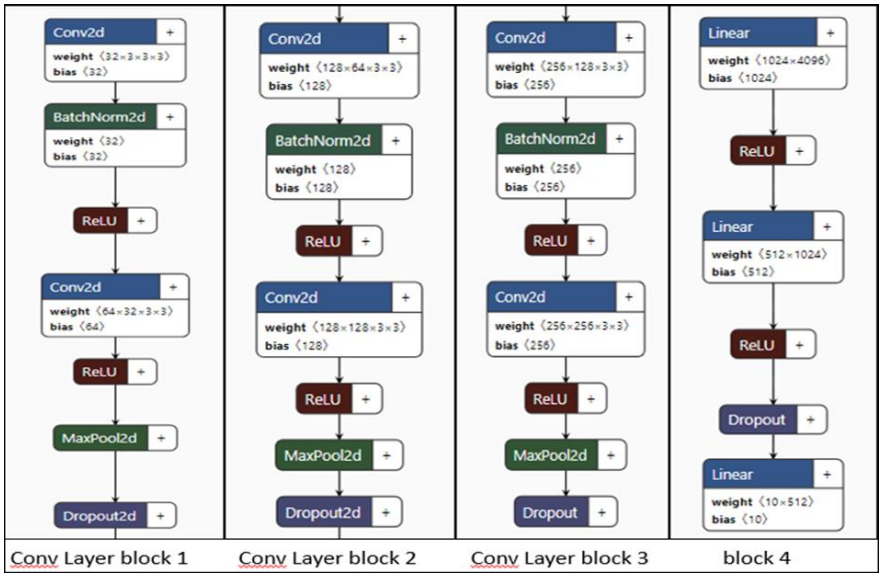


Abbildung 5-1: Architektur Basis-CNN

## 5.2 Anpassung der Hyperparameter und Ergebnisse

In der Tabelle 4-1 ist einer Übersicht der Hyperparameter zu finden, welche an den verschiedenen CNNs eingestellt wurden.

Tabelle 5-1: Übersicht der Hyperparameter der MLP-Netze

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **Lernrate** | 0,001 | 0,001 | 0,001 | 0,001 | 0,00175 |
| **Batchgröße** | 512 | 512 | 256 | 512 | 512 |
| **Optimizer** | SGD | SGD | Adam | SGD | SGD |
| **Dropout** | 50%? | 50% | 50% | 50% | 50% |
| **Neuronenzahl** | 198154 | 801792 | 198154 | 288746 | 198154 |
| **Konvolutionsschichten** | 6 | 6 | 6 | 9 | 6 |
| **Poolingschichten** | 3 | 3 | 3 | 3 | 3 |
| **Fully-Connected-Schichten** | 2 | 2 | 2 | 6 | 2 |

Beim CNN entspricht das Basisnetz auch wieder dem Netz aus der ersten Spalte

**Learningrate:** Die Lernrate bei einem CNN muss ebenfalls wie bei einem MLP auf jedes individuelle Netz angepasst werden. Wählt man diesen Faktor zu groß ist das Netz nicht lernfähig. Wird dieser jedoch zu klein gewählt kann das Netz ebenfalls nicht richtig lernen und die Lerndauer wird drastisch erhöht. Die zeigt der Verlauf der Accuracy der Lern- und der Validierungsdaten. Die Abbildung 5-2 zeigt der Verlauf der Accuracy der Lern- und der Validierungsdaten.

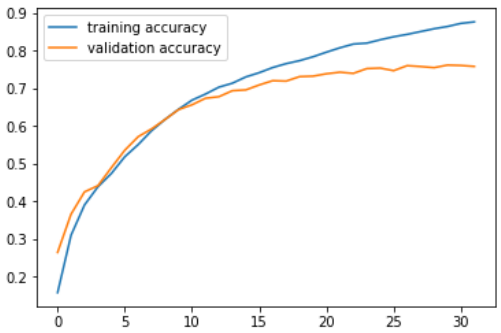


Abbildung 5-2: Verlauf der Accuracy bei Änderung der Learningrate

**Batchsize:** Auch bei dem CNN hat sich gezeigt, dass es bei einer falschen Kombination von Learning Rate und Batchsize zu einem fehlerhaften Modell kommt. Wird jedoch eine feste Learning Rate wie im Ausgangs-CNN von 0,001 angenommen, so zeigte sich wie in Abbildung 5-3 dargestellt, dass die hierzu passende Batch Size 512 ist.

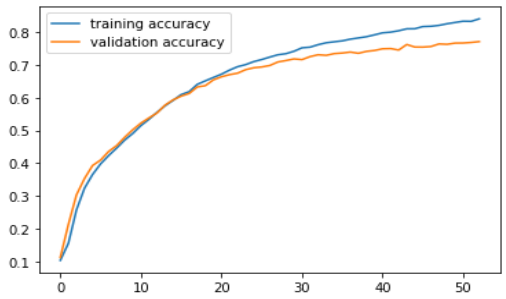


Abbildung 5-3: Verlauf der Accuracy bei Änderung der Batchsize

**Optimizer:** Das neuronale Netz mit dem Verlauf der Accuracy für die Lern- und Validierungsdaten aus der Abbildung 5-4 verwendet den Optimizer Adam mit einer Learning Rate von 0.001. Durch den Einsatz des Adam-Optimizers konnte dieses Netz die beste Accuracy mit den Validierungsdaten erzielen, nach 10 Epochen konnte ungefähr eine Genauigkeit von 60% erreicht werden. Bei dem CNN hat sich gezeigt, dass es bei einer falschen Kombination von Learning Rate und Optimizer zu einem fehlerhaften Modell kommt.

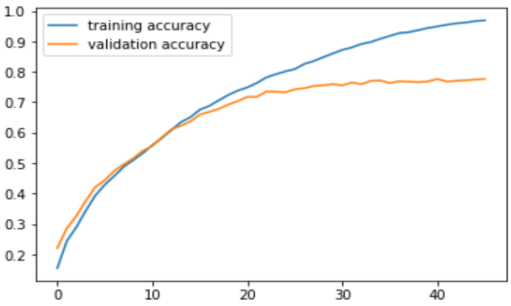


Abbildung 5-4: Verlauf der Accuracy bei Änderung der Optimizer

**Neuronenzahl und Dropout:** Anders als beim MLP Netz besitzt das CNN eine wesentlich tiefere Struktur und wirkt sich damit auch sehr stark auf die gesamte Neuronenzahl auf. Um das beste Ergebnis zu finden wurde im ersten Testlauf mit einer sehr geringen und im zweiten Testlauf mit einer sehr hohen Neuronenzahl gearbeitet. Im Zuge der Parameteranpassung wurden die Neuronen, nachdem zuvor beschrieben Schema weiter angepasst, um ein optimales Ergebnis, welches in Abbildung 5-5 zu erkennen ist, zu erreichen. Vergleichbar mit Neuronzahländerung des MLP Netzes wurde auch hier jede Parameteränderung mit und ohne Dropout 20% untersucht. (Siehe Tabelle 5-2)

Tabelle 5-2: Neuronenanzahl der einzelnen Schichten

| Conv.-Layer Block 1 | Conv.-Layer Block 2 | Conv.-Layer Block 3 | FC-Layer 1 | FC-Layer 2 | FC-Layer 3 |
|---|---|---|---|---|---|
| 143360 | 271360 | 491520 | 43200 | 6000 | 1210 |

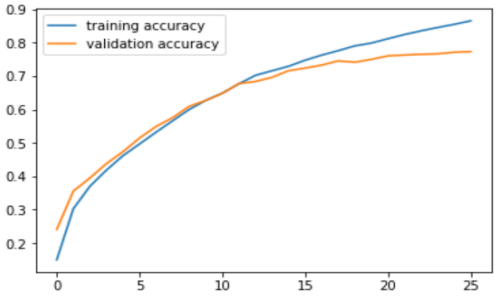


Abbildung 5-5: Verlauf der Accuracy bei Änderung der Neuronenzahl

**Schichtenzahl:** Das längste CNN besitzt drei Blöcke aus drei Konvolutionsschichten gefolgt von einer Poolingschicht. Anschließend folgen sechs Fully-Connected-Schichten. So wie beim sehr tiefen MLP dauerte auch hier der Lernvorgang durch die hohe Parameteranzahl sehr lange. Außerdem liegen auch hier, analog zum tiefen MLP, Schwingungen in der Accuracy der Trainings- und der Validierungsdaten vor. Im Gegensatz zum langen MLP erzielt dieses CNN schlechte Ergebnisse bei der Klassifikation. Die Abbildung 5-6 zeigt den Verlauf der Accuracy für die Lern- und für die Validierungsdaten für das lange CNN.

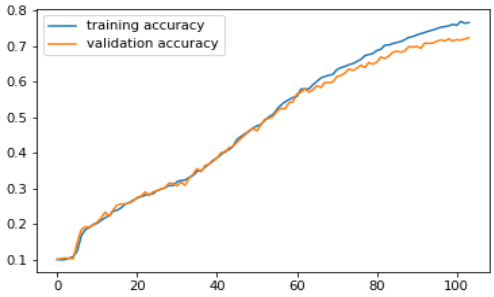


Abbildung 5-6: Verlauf der Accuracy bei Änderung der Schichtenzahl

Bei dem CNN sind deutlich bessere Validierungsdaten erkennbar im Vergleich zu den MLP-Netzen. Jedoch ist auch bei allen CNN-Netzen sowohl die Parameterzahl als auch die Laufzeit dafür angestiegen. Die Versuche haben gezeigt, dass das CNN mit dem ADAM-Optimizer die besten Ergebnisse liefert.

Tabelle 5-3: Ergebnisse der CNNs in Abhängigkeit der Hyperparameter

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **Epochenzahl** | 54 | 27 | 201 | 105 | 33 |
| **Laufzeit [s]** | 863 | 3632 | 7246 | 2368 | 1136 |
| **Accuracy Trainingsdaten [%]** | 84,1 | 86,5% | 94,7 | 76,7% | 86,36 |
| **Accuracy Validierungsdaten [%]** | 77,1 | 77,4 | 79,5 | 72,4% | 76,12 |
| **Parameteranzahl (·10³)** | 5851 | 252886 | 5851 | 6244 | 5851 |

Wie schon bei dem MLP-Netz ist wieder zu erkennen, dass sowohl Hunde als Katzen erkannt wurden und auch andersrum. Ansonsten ist auffällig, dass die Klassifizierung wesentlich besser abgelaufen ist. Die Unterscheidung von Autos und Trucks hat diesmal besser stattgefunden. (siehe Abbildung 5-7)

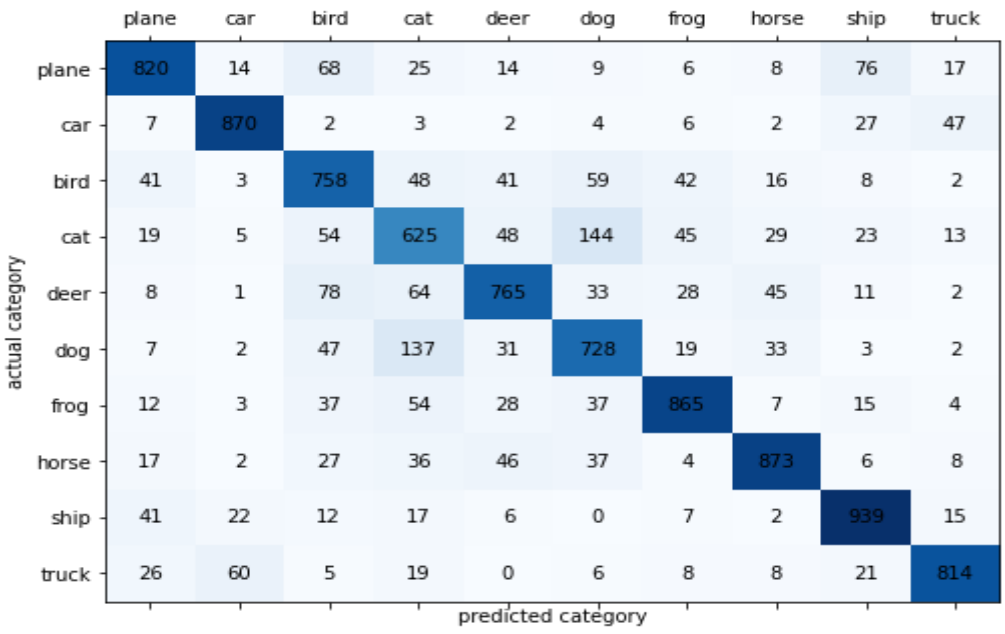


Abbildung 5-7: Confusion-Matrix für das beste MLP mit dem Adam-Optimizer

# 6 Transfer Learning

Unter Transfer-Learning versteht man das Anpassen bereits existierender neuronaler Netze auf eine neue Aufgabe. Hierbei wird oftmals die letzte Schicht des vorhandenen Netzes auf das aktuelle Problem angepasst. Somit muss das Netz lediglich die Output-Beziehungen neu erlernen, da die vorhergehenden Schichten bereits erlernt haben, Formen bzw. Zusammenhänge zu erkennen. Hiermit kann die Programmierarbeit deutlich reduziert und gleichzeitig das Ergebnis maximiert werden, da eine komplexe Struktur bereits vorhanden ist. Um eine möglichst gute Vergleichbarkeit zu schaffen, wurden verschiedene Netze gegeneinander verglichen, indem sie mit dem neuen Datensatz bestückt wurden. (Tabelle 6-1)

Tabelle 6-1: Übersicht der verwendeten Netze

| | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| Eingesetztes Netz | VGG-16 | Shuffle Net | ResNet18 | AlexNet | GoogLeNet |

**AlexNet:** Das CNN AlexNet gewann 2012 den Bildklassifizierungswettbewerb ImageNet mit einer Top-1-Fehlerrate von 15,3% und einer Top-5-Fehlerrate von 26,2%. Das Netz besteht aus fünf Konvolutions- und drei Fully-Connected-Schichten. Durch AlexNet wurden neue Innovationen im Bereich der CNNs geschaffen. Eine hiervon war die Verwendung der ReLU-Funktion statt der tanh-Funktion, wodurch der Lernprozess beschleunigt wurde. Zudem wurde AlexNet auf zwei GPUs trainiert, da das Training für damalige Verhältnisse zu viel Speicherplatz in Anspruch nahm. Die dritte Innovation war das sogenannte Overlapping Pooling, bei dem der Pooling-Filter mit einem Stride verschoben wird, der kleiner als der Filter selbst ist. Hierdurch konnte Overfitting vermieden werden. Die Architektur von AlexNet ist in der Abbildung 6-1 skizziert. [4]

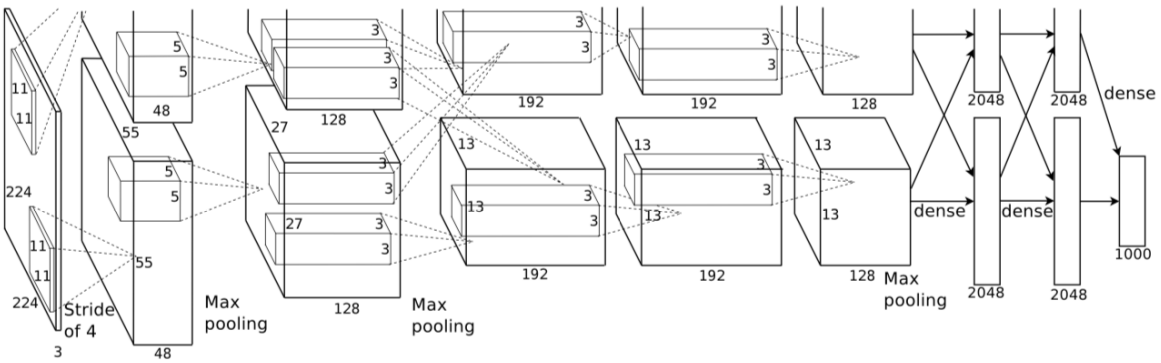


Abbildung 6-1: Architektur von AlexNet

Nachfolgend in Abbildung 6-2 ist die Accuracy von AlexNet dargestellt.

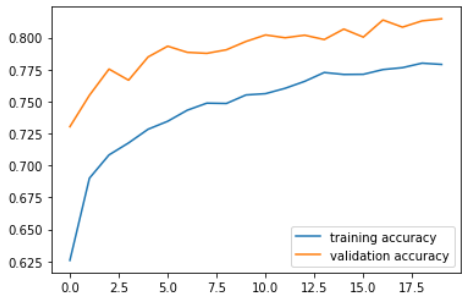


Abbildung 6-2: Verlauf der Accuracy bei AlexNet

**VGG-16:** Das VGG-16 Netz wurde von K. Simonyan und A. Zisserman von der Universität Oxford veröffentlicht. Das Modell erreichte eine Genauigkeit von 92,7% bei dem ImageNet-Datensatz und war damit einer der besten fünf Netze für diesen Datensatz.

Nachfolgend ist die Architektur dargestellt (Abbildung 6-3). Das VGG-16 baute dabei auf AlexNet auf und ersetze dabei die große Kernel-Size der Filter (11x11 im ersten Layer und 5x5 im zweiten Layer) durch 3x3-Kernel-Size Filter.

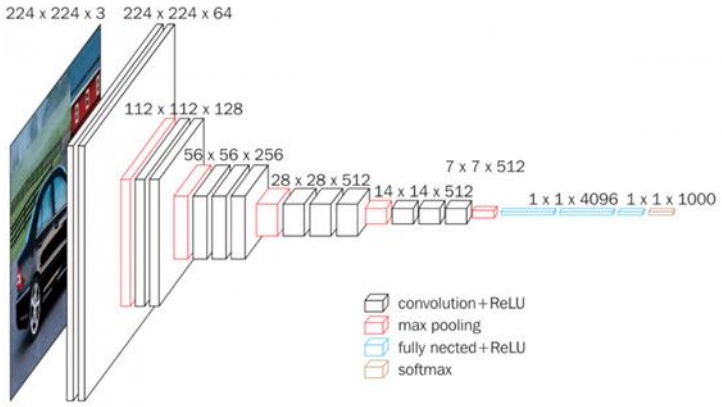


Abbildung 6-3: Architektur von VGG-16

Durch das Neutrainieren der letzten Schicht des Netzes auf den CIFAR-10 Datensatz brachte das Netz eine Genauigkeit von ca. 83%, wie in Abbildung 6-4 dargestellt. [5]

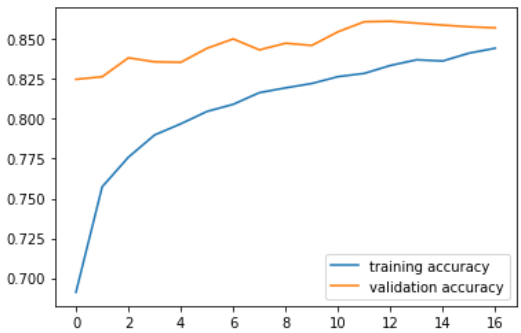


Abbildung 6-4: Verlauf der Accuracy bei VGG-16

**ResNet18:** Sehr tiefe neuronale Netze besitzen das Problem eines verschwindenden Gradienten. Dies bedeutet das der Einfluss der Parameter in den ersten Schichten auf das

Endergebnis beim Training nicht mehr ermittelbar ist. Residual Neuronal Networks besitzen sog. Identity-Shortcut-Connections, sprich Verknüpfungen zwischen einzelnen Neuronenschichten. Die Idee besteht darin, dass Schichten nur die Differenz zur Vorgängerschicht erlenen müssen, sodass das Problem des verschwinden Gradients durch einen direkten Pfad auf tiefe Schichten behoben. [6]

Die Ergebnisse aus dem gelerntem ResNet sind nachfolgend (Abbildung 6-5) dargestellt.

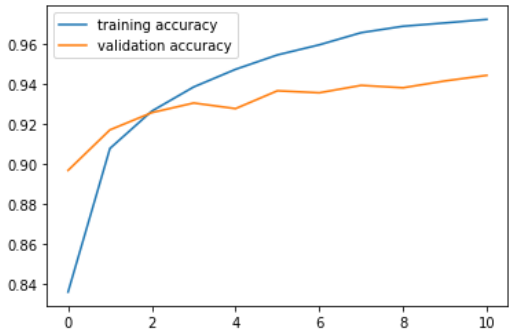


Abbildung 6-5: Verlauf der Accuracy bei ResNet18

**ShuffleNet:** Das ShuffleNet ist besonders durch gleichmäßigen Durchmischung nach der Gruppierten Faltung erkennbar. (Eingabe-Features werden entlang der Kanaldimensionen in verschiedene Gruppen unterteilt, um im Anschluss die Faltungsoperation durchführen zu können.) In der nachfolgenden Abbildung 6-6 wird grafisch der Vergleich zwischen (a) einer typischen Gruppierung im Conv-Layer wie sie bei einem AlexNet oder ResNet zu finden ist und (b)-(c) ein Shuffle Struktur, wie sie im ShuffleNet zu finden ist, dargestellt. Die gleichmäßige Durchmischung nach der Gruppierung hat den Vorteil, dass die Ausgangskanäle mit einer Vielzahl an zufälligen Eingangskanälen in Verbindung steht und somit die globale Informationszirkulation verbessert wird. [7]

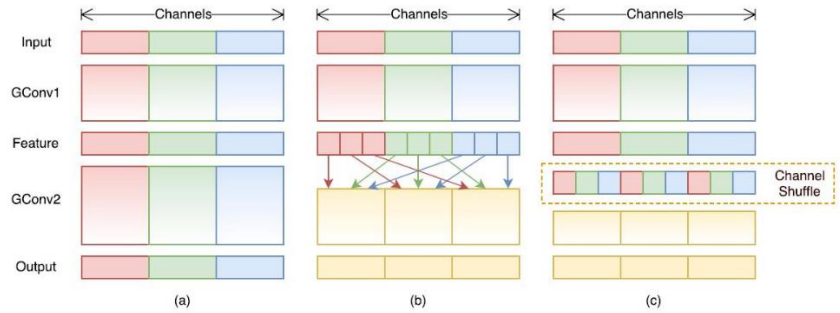


Figure 1: Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.

Abbildung 6-6: Architektur SchuffleNet

Nachfolgend ist der zum ShuffleNet zugehörige Verlauf der Accuracy dargestellt. (Abbildung 6-7)

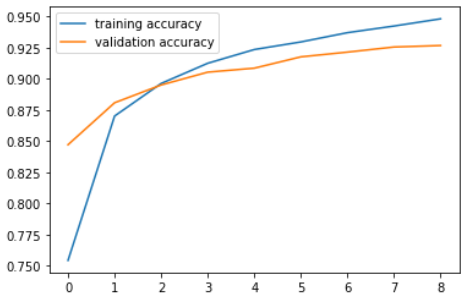


Abbildung 6-7: Verlauf der Accuracy bei ShuffleNet

**GoogLeNet:** Bei GoogLeNet handelt es sich um ein CNN Netz, welches 22-layer besitzt. Es wurde von Programmierern der gleichnamigen Firma Google entwickelt. 2014 gewann dieses Netz bei einem Contest (ILSVRG) mit einer Fehlerrate von lediglich 6.7%. Die untenstehende Abbildung 6-9 zeigt den Aufbau des neuronalen Netzes. Das interessante an dieser Architektur ist , dass dieses Netz eine bedeutend höhere Genauigkeit als der Gewinner von 2012 aufweist, hierbei jedoch nur ein zwölftel der Parameter benötigt.

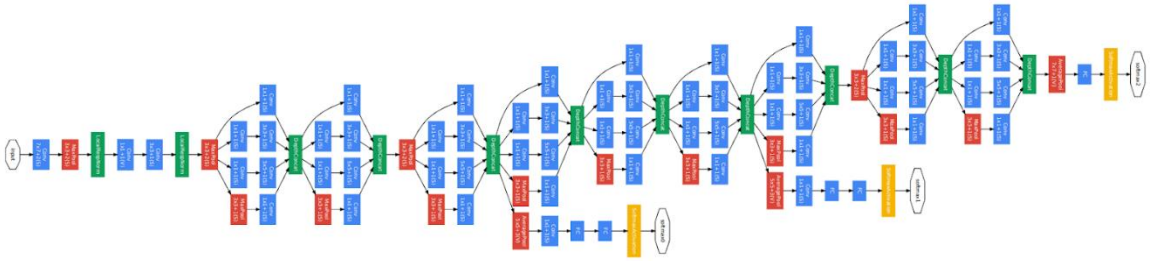

Abbildung 6-8: Architektur GoogLeNet

Accurancy googlenet

Bei den Transfer-Learning-Netzen ist zu erkennen (Tabelle 6-2), dass die Epochen nun alle deutlich geringer sind, dafür jedoch die Zeit auch höher. Dies resultiert unter anderem auch aus den hohen Parameterzahlen. Durch das Transfer-Learning wurden deutlich bessere Ergebnisse der Accuracy im Vergleich zu den vorherigen Netzen erreicht.

Tabelle 6-2: Ergebnisse des Transfer Learnings

| | **A** | **B** | **C** | **D** | **E** |
|---|---|---|---|---|---|
| Epochenzahl | 18 | 9 | 12 | 21 | 14 |
| Laufzeit [s] | 11653 | 2744 | 3794 | 6388 | 5744 |
| Accuracy Trainingsdaten [%] | 83,4 | 94,4% | 97,2 | 78% | 97,1 |
| Accuracy Validierungsdaten [%] | 86,1 | 92% | 94,4 | 81,4% | 94,3 |
| Parameteranzahl ($\cdot 10^6$) | 119,6 | 1,28 | 11,2 | 54,6 | 5,6 |

Wie schon bei dem MLP-Netz ist wieder zu erkennen, dass sowohl Hunde als Katzen erkannt wurden und auch andersrum. Ansonsten ist auffällig, dass

die Klassifizierung wesentlich besser abgelaufen ist. Die Unterscheidung von Autos und Trucks hat diesmal besser stattgefunden. (Abbildung 6-9)

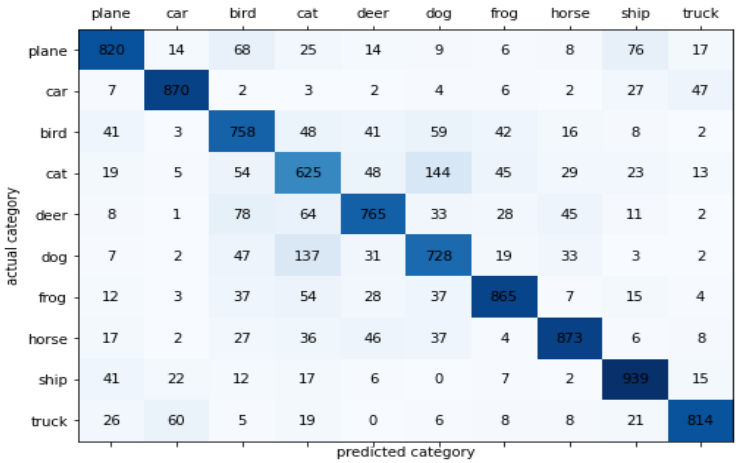


Abbildung 6-9: Confusion-Matrix ResNet-18

# 7 Fazit

Neuronale Netze sind ein mächtiges Instrument, um Strukturen beziehungsweise Muster zu erkennen und auszuwerten. Hierbei wird jedoch das Netz lediglich von außen betrachtet und eine Prognose über die Ergebnisse kann bestmöglicherweise lediglich abgeschätzt werden. Prinzipiell kann jedoch gesagt werden, dass für einfache, einschichtige Muster ein MLP meist ausreichend ist.

Um komplexere Zusammenhänge auszuwerten, ist ein CNN besser geeignet. Speziell bei der Bilderkennung zeigen sich hierbei deutliche Vorteile des CNNs, da bei der Auswertung des Bildes die Informationen der umliegenden Pixel nicht verloren gehen.

In den zuvor gezeigten Beispielen zeigt sich, dass bei dem MLP lediglich eine Genauigkeit von 55% erreicht wird. Des Weiteren werden bei manchen Klassen überdurchschnittlich viele Kategorien zugeordnet, oder es fällt eine Klasse komplett weg und wird einfach nicht erkannt (Keine richtige oder auch falsche Zuordnung für eine Klasse). Bei einem CNN kann hierbei die Genauigkeit bereits auf 80% erhöht werden. Dies zeigt deutlich die Überlegenheit eines CNN Netzes gegenüber eines MLPs.

Um die höchste Genauigkeit zu erreichen, wird Transfer-Learning angewendet. Hierbei lässt sich eine Genauigkeit von 94,4% erreichen. Durch die Verwendung der bereits hoch entwickelten und trainierten Netze, lassen sich eine schnellere Erstellung und eine bessere Modellqualität realisieren. Hierbei wird jedoch eine höhere Rechenkapazität benötigt.

# Literatur- und Quellenverzeichnis

[1] J. Theuerkauf, Schreiben im Ingenieurstudium. Effektiv zu Bachelor-, Master- und Doktorarbeit, Paderborn: utb, 2012.

[2] Name Nachname, „Titel,“ in *Buchtitel*, Ort, Verleger, Jahr, p. Seitenbereich.

# Anhang A: Überschrift

# Anhang B: Überschrift