Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Bachelorarbeit Kognitionswissenschaft

Pflanzenklassifizierung mittels tiefer neuronaler Netze

Paul Schmidt-Barbo

18.04.2017

Gutachter

Prof. Dr. Hanspeter A. Mallot
Cognitive Neuroscience Department of Biology
Universität Tübingen

Betreuer

Gerrit Ecke
Cognitive Neuroscience Department of Biology
Universität Tübingen

Schmidt-Barbo, Paul
Pflanzenklassifizierung mittels tiefer neuronaler Netze
Bachelorarbeit Kognitionswissenschaft
Eberhard Karls Universität Tübingen
Bearbeitungszeitraum: 18.04.2017-18.08.2017

Zusammenfassung

Über die letzten Jahre hinweg bilden tiefe neuronale Netze einen aufstrebenden Zweig in der maschinellen Erkennung natürlicher Bilder. Sie erzielten beachtliche Erfolge in Wettbewerben wie ImageNet [Russakovsky et al., 2012] oder PlantClef [Clough and Sanderson, 2003] und bilden den heutigen State of the Art.
Gut funktionierende Netzwerke schienen maßgeblich durch ihre Tiefe bestimmt zu werden, was die Entwicklung immer tiefere Architekturen vorantreibt. Das Verständnis über die genau Funktionsweise des Netzwerks jedoch bleibt bei solch tiefen Architekturen oftmals auf der Strecke.
In unsere Arbeit nehmen wir uns dieser Problematik an und versuchen Erkenntnisse über die interne Vorgänge zweier bekannter Netzwerke bei der Klassifizierung von Pflanzen zu erlangen. Der erste Teil unserer Arbeit bildet dabei eine Replikation im Rahmen des PlantClef Wettbewerbes 2016. Wir repizierten die Arbeit von Mostafa Mehdipour Ghazi und seinen Kollegen[Ghazi et al., 2016] , welche ein Ensemble aus Netzwerken zur Klassifizierung von Pflanzen verwendeten. Zum Einsatz kommen dabei das Netzwerke von Koren Simonyan und Andrew Zisserman [Simonyan and Zisserman, 2014] sowie das von Christian Szegedy, Wei Liu et al. [Szegedy et al., 2014]. Im zweiten Teil untersuchen wir beide verwendeten Netzwerkarchitekturen und visualisieren ihre internen Vorgänge bei der Pflanzenerkennung. Aus zeittechnischen Gründen widmeten wir uns vermehrt dem Netzwerk von Christian Szegedy. Auf die Frage, ob eine taxonomische Klassifizierung über die Tiefe der Netzwerke zu erkennen ist, ließen wir das Netzwerke nach unterschiedlichen Schichten Vorhersagen über die Pflanzenspezies, Pflanzengattung und Pflanzenfamilie treffen. Wir konnten zeigen, dass für alle Ordnungen die Vorhersagen nach der tiefsten Schicht am genausten waren. Somit schließen wir, dass das Netzwerk keine taxonomische Klassifizierung der Pflanzenbilder über die Tiefe der Schichten vornimmt.
Im weiteren visualisierten wir mithilfe einer Auffaltung (Deconvolution) der Netzwerkarchitektur [Zeiler and Fergus, 2013] die Aktivierungsmuster einzelner Schichten. Wir können zeigen, dass das Netzwerk lernt pflanzenspezifische Merkmale in den Bildern zu erkennen, welche auch als Bestimmungsmerkmale in der Botanik dienen. Das Netzwerk scheint sich vor allem auf Blatt- und Blütenform zu spezialisieren, was wir auf ihren Informationsgehalt zurückführen. Deutlich wird jedoch auch, dass keine frühe Einteilung in Pflanzenorgane stattfindet, sondern sich organunabhängige Filter bilden. Eine Generalisierung und ein Transferlernen auf Genus- und Familienklassifikation war möglich, was für allgemein gültige pflanzenspezifische Merkmalsdetektoren spricht.

Danksagung

Ich bedanke mich bei meinem Betreuer Gerrit Ecke, der immer einen Ansprechpartner darstellte. Ebenso möchte ich einen besonderen Dank an Maximilian Beller aussprechen, der mich tatkräftig bei meiner Arbeit unterstützt und dadurch vieles erst ermöglichte.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Sichwortverzeichnis

Low-, Mid-, & High-Vision	Einteilung der visuellen Verarbeitung. Von der Umwandlung eines Lichtreizes in ein Rezeptorpotential, über die Detektion von Kanten, bis hin zur Objekt Segmentierung und Objekterkennung.
Simple & Complex Cells	Zelltypen des visuellen Kortex, welche auf die räumliche Aufteilung von Reizen selektiv reagieren.
Gabor Filter	Mathematisches Modell der Simple Cells.
Convolution	Mathematische Operation, die als Gewichtung eines Signals $g(x)$ mit einem Signal $h(x)$ verstanden werden kann.
Deconvolution	Verfahren zur Umkehrung der Faltungsoperation.
KNN	Künstliche Neuronale Netze.
CNN	Convolutional Neural Network.
Perceptron	Vereinfachtes knstliches neuronales Netz von F. Rosenblatt [Rosenblatt, 1958].
ReLU	Rectified Linear Unit. Definiert durch: $max(0, x)$.
Conv Layer	Concolutional Layer. Schichten in denen eine Faltungsoperation stattfindet.
Pooling Layer	Schichten, die die räumliche Auflösung durch Pooling Funktionen (z.B. max) reduzieren.

FC Layer	Fully-Connected Layer. Vollständig mit dem nachfolgenden Layer verbundene Neuronenschicht.
Feature-Map	Durch Convolution der Eingabe mit einem bestimmten Filter ausgelöste neuronen Aktivität.
Batch	Gruppierte Eingabedaten.
Overfitting	Überanpassung des Netzes auf den Trainingsdatensatz.
Hebbsche Lernregel	$\triangle w_{i,j} = \theta \cdot a_i \cdot o_j$
Backpropagation	Überwachtes Lernverfahren, welches den Fehler rückwärts durch das Netzwerk propagiert.
Momentum	Erweiterung des Backpropagationterms durch Addieren der vorherigen Gewichtsanpassung.
Weight-Decay	Erweitung des Backpropagationterms durch stetiges verringern der Gewichte.
Drop-Out	Regularisierungsmethode, bei der mit einer vordefinierten Wahrscheinlichkeit Neuronen inaktiv bleiben.
Teaching Signal	Gewünschte Ausgabe mit der die Ausgabe des Netzes verglichen wird.
Kreuzentropie Fehlerfunktion	$-\frac{1}{N}\sum_{img}\left[P(Label=i)log(P(\hat{Label}=i)) + (1-P(Label=i)log(1-P(\hat{Label}=i))\right]$
Vanishing Gradient Problem	Von Layer zu Layer immer schwächer werdendes Fehlersignal, was dazu führt dass Gewichtsupdates immer kleiner werden oder ausbleiben.
Ground Truth	Als wahr oder richtig geltende Klasse der Daten.
Top-N-Fehler	Maß zur Evaluation eines Netzwerkes. Prüft ob die N Klassen mit den höchsten Ausgabewerten das wahre Label enthalten.
ILSVRC	ImageNet Large Scale Visual Recognition Challange

VOC	Visual Object Classes Challenge
CLEF	Conference and Labs of the Evaluation Forum
MAP	Mean Average Precision
Vgg16	Name der Netzwerkarchitektur von Koren Simonyan und Andrew Zisserman [Simonyan and Zisserman, 2014].
GoogLeNet	Name der Netzwerkarchitektur von Christian Szegedy, Wei Liu et al. [Szegedy et al., 2014].
IncpetionV1	Name unter dem eine abgewandelte Version des GoogleNet bekannt wurde. 5×5 in den Inception Modulen wurden durch 3×3 Filter ersetzt
AlexNet	Name des Netzwerks von Alex Krizhevsky, Geoffrey Hinton et al. [Krizhevsky et al., 2012].
LeNet	Name des Netzwerks von Yann LeCun, L'eon Bottou et al. [Lecun et al., 1998].
Epoche	Der Forward- und Backwardpass für alle Trainingsdaten.

Kapitel 1

Einleitung

Das menschliche Sehsystem liefert sensorische Informationen, die es uns ermöglichen sicher im Alltag zu agieren. Diese Fähigkeit beruht nicht nur auf einer reinen Detektion von Lichtreizen, sondern insbesondere auf ihrer Verarbeitung und bewussten Wahrnehmung.
Bereits in den frühen Kindertagen [Johnson et al., 1991] besitzen wir die Fähigkeit Szenen zu Segmentieren und Objekte zu Klassifizieren. Dabei stellen der Blickwinkel, die Zusammengehörigkeiten oder Okklusion von Objekten keine Probleme für uns dar. Schon beim Eintreffen auf der Netzhaut werden Farben, Kontraste oder Bewegungen analysiert. Grundlegende Verarbeitungsschritte, die als *low-level Vision* verstanden werden. Spätere Abschnitte nutzen diese extrahierten Informationen, um daraus komplexere Eigenschaften des Signals, wie die Objekt Zugehörigkeit zu schließen und letztendlich Objekte zu erkennen. Diese Vorgänge fallen unter die Begriffe *mid-* und *high-level Vision*.
Während für die grundlegende Merkmalsanalyse bereits Modelle existieren, stellen mid- und high-level Vision Computer basierende Verfahren immer noch vor große Herausforderungen.
Hubel und Wiesel [Hubel and Wiesel, 1962] konnten 1962 zeigen, dass der visuelle Kortex kleine Zellregionen besitzt, welche auf spezifische Bereiche des visuellen Feldes reagieren. In ihren Experimenten [Hubel and Wiesel, 1959] präsentierten sie Katzen einen Lichtbalken in variierender Orientierung. Die Präsentationen erfolgten über das Sichtfeld verteilt, während zeitgleich bestimmte Neuronen des visuellen Kortex abgeleitet wurden. Die abgeleiteten Neuronen zeigten orientierungs- , orts- und auch bewegungsspezifische Antwortmuster. Ebenso erkannten Hubel und Wiesel [Hubel and Wiesel, 1962], dass ihre rezeptiven Felder in exzitatorische und inhibitorische Regionen abgegrenzt werden konnten. Die Antworten schienen sich linear zu summieren, wenn die Stimulation räumlich oder zeitlich zunahm. Bekannt wurden diese Zellen unter dem Namen *Simple Cells*.
Im Gegensatz dazu standen die sogenannten *Complex Cells*, welche ebenfalls

von Wiesel und Hubel [Hubel and Wiesel, 1962]entdeckt eine räumliche Invarianz besitzen. Ihre rezeptiven Felder lassen sich nicht wie bei den Simple Cells räumlich einteilen, sondern sind unabhängig von der genauen Position eines Reizes. Ihr Eingaben erhalten sie aus einer Reihe von Simple Cells, welche sie integrieren und aufsummieren.
Beide Zelltypen werden als lineare Operatoren angesehen und können mit Hilfe des mathematische *Gabor Model* beschrieben werden [Gabor, 1946, Marĉelja, 1980]. Ein Bild wird dabei als Überlagerung einzelner sinusförmiger Wellen mit bestimmten Frequenz betrachtet. Jede vorher als Pixel gehandhabte Position im Bild, gibt nun Auskunft über die Frequenz und Orientierung der Welle. Eine Gauß-Funktion dient als Filter Maske für bestimmte Frequenzen im Bild. Durch Faltung (*Convolution*) über das Bild, kann so das Reaktionsmuster eines Neurons auf bestimmte Bildmerkmale anhand der Ausgabewerte des Gabor-Filters modelliert werden.
Auf dieser Idee aufbauend, entwickelte sich das Konzept der Convolutional Neural Networks (CNN). Vorwärts gerichtete künstliche Neuronale Netze mit spezialisierten Komponenten zum erkennen bestimmter Bildeigenschaften (*Features*). Anwendung finden diese heutzutage vornehmlich in Bereichen der Bildverarbeitung und Objekterkennung, wo sie über die letzten Jahre große Erfolge erzielen konnten.

In unsere Bachelorarbeit befassen wir uns mit einem System zur Pflanzenklassifizierung, welches den zweiten Platz in der letztjährigen PlantClef Challange [Goau et al., 2016] erzielen konnte. Neben der Replikation der Daten steht dabei die Analyse interner Vorgänge im Vordergrund.

Der erste Abschnitt fokussiert sich auf die Ergebnisse des Wettbewerbs im Detail 3. Hier werden wir die Strukturen der Netze und die gewählten Implementationsansätze näher erläutern. Darauf folgt im zweiten Abschnitt eine Versuch Erkenntnisse über Funktionsweise der Netze zu erlangen. Mit Hilfe eines Verfahrens von Zeiler und Fergus [Zeiler and Fergus, 2013, Zeiler et al., 2011] versuchen wir Berechnungen zu visualisieren und Rückschlüsse auf mögliche rezeptive Felder zu ziehen. Die Frage die sich dabei stellt ist, inwieweit sich pflanzenspezifische Merkmalsdetektoren gebildet haben. Des weiteren untersuchen wir die Vorhersagekraft der Netze in Abhängigkeit zur Netzwerktiefe auf eine erkennbare Taxonomie. Ziel ist es zu prüfen, ob die Vorhersage in einzelnen Schichten mit der Systematik im Pflanzenreich korrespondiert.

Bevor wir uns der genauen Anwendung und Analyse Convolutional Neural Networks zuwenden, wollen wir zunächst einige grundlegende Begriffe und Funktionsweisen erläutern.

Kapitel 2

Grundlagen und Exkurs

Die Convolutional Neural Networks (CNN) gehören zur Kategorie der vorwärts gerichteten künstlichen Neuronalen Netzen. Dabei lassen sich künstliche Neuronale Netze (KNN) durch drei grundlegenden Elemente erklären.

i. Aktivierung und Aktivitätsausbreitung der Neurone

ii. Synaptische Verbindungen samt dynamischer Anpassung

iii. Topologie und Anordnung der vernetzten Neuronen

2.1 Künstliche Neuronale Netze (KNN)

2.1.1 Aufbau und Architektur

Abbildung 2.1 zeigt eine einzelnes Neuron j. Es enthält gewichtete Eingaben $w_{ij} \cdot x_{ij}$ aus weiteren Neuronen oder externen Quellen. Alle eintreffenden Eingaben bilden summiert das Potential des Neurons u_i. Jedes Neuron besitzt eine Aktivierungsfunktion, welche aus dem vorhandenen Potential u_i die Aktivität des Neurons $f(u_i)$ berechnet. Üblicherweise werden nicht-lineare Funktionen wie *Sigmoid*, *tanh* oder *ReLU (Rectified Linear Unit)* verwendet. Grund dafür ist, dass durch eine nicht-lineare Funktion auch nicht-lineare Eigenschaften der Eingabe dargestellt werden können.

2.1.2 Synaptische Verbindung

Die Gewichte der Verbindungen werden durch Lernregeln angepasst. Änderung sind dabei abhängig vom aktuellen Aktivierungszustand des Neurons. Neben unüberwachtem Lernen, welches auf der Hebbschen Lernregel [Hebb, 1949] basiert, sind es die Überwachten Lernverfahren, die uns im weiteren Verlauf

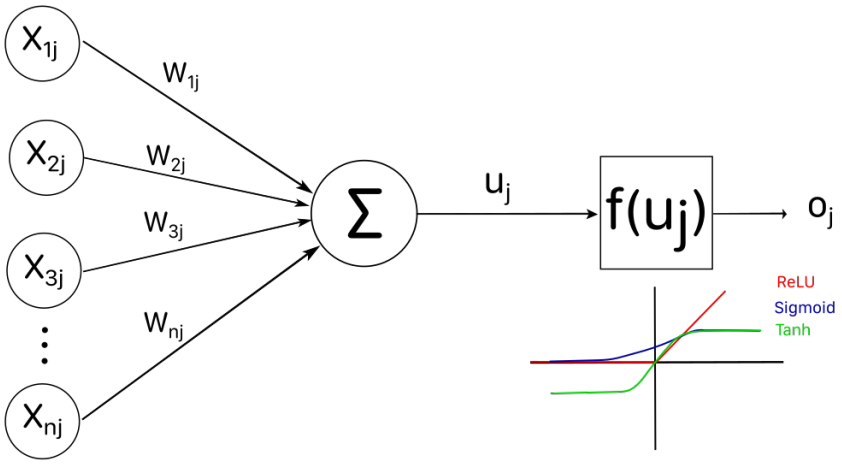


Abbildung 2.1: Aktivierung eines *Perceptrons* [Rosenblatt, 1958]. Ein einfaches Künstliches Neuronales Netz (KNN), bestehend aus einem einzigen Neuron.

interessieren. Dabei existiert ein Lernsignal (*Teaching Signal*) welches das Neuron entsprechend einer gewünschten Ausgabe anpasst.

2.1.3 Topologie

Die Topologie eines künstlichen neuronalen Netzes (KNN) ist bestimmt durch seine Anordnung einzelner Neuronen. In Schichten (*Layer*) gruppiert, lassen sich drei Typen unterscheiden:

1. Input Layer: Die Neuronen dieser Schicht erhalten externe Informationen und leiten sie ohne weitere Berechnungen weiter.

2. Hidden Layer: Die verdeckten Neuronen (*Hidden Nodes*) verbinden die Eingabeschicht mit der Ausgabeschicht. Ein Netzwerk kann keine (*Single Layer Perceptron*) oder mehrere dieser Schichten (*Multi Layer Perceptron*) enthalten. Sie bestimmen die Tiefe eines Netzes.

3. Output Layer: Diese Schicht liefert Informationen über die Berechnungen und Ausgaben des Netzes.

In vorwärts gerichteten (engl. feedforward) Netzen wird die Information von der Eingabeschicht zur Ausgabeschicht weitergetragen. Es existieren keine Zyklen und keine Querverbindungen. Während feedforward Netze ohne Hidden Layer nur lineare Funktionen darstellen können, ermöglichen Hidden Layer auch nicht-lineare Funktionen zu lernen. Bekannteste Beispiele für solche Funktionen sind das Klassifizieren von Datenpunkten.

2.1.4 Backpropagation

Gelernt wird eine solche nicht-lineare Funktion durch ein Verfahren, welches als Backpropagation [Werbos, 1974, Rumelhart et al., 1986] bekannt wurde. Es fällt in die Kategorie der Überwachten Lernverfahren und basiert auf dem Prinzip der Fehlerminimierung.
Zu Beginn der Trainingsphase werden alle Gewichte im Netz zufällig initialisiert und eine Eingabe durch das Netz propagiert (*Forwardpass*). Die Ausgabe wird mit der gewünschten Ausgabe (*Teaching Signal*) verglichen. Die Differenz wird durch eine Fehlerfunktion E (*Loss Function*) berechnet. Der Fehler wird nun von der Ausgabeschicht zurück zur Eingabeschicht geschickt und sorgt auf seinem Weg für die Anpassung der Gewichte(*Backwardpass*). Gewichtsänderungen werden dabei Abhängig vom Einfluss des jeweiligen Gewichts auf den berechneten Fehler vorgenommen. Ziel ist es jedes Gewicht w_i so anzupassen dass die Fehlerfunktion ihr Minimum erreicht. Dies geschieht durch Abstieg in Richtung des Gradienten der Fehlerfunktion bezüglich des Gewichts w_i. Gewichtsupdates ergeben sich dabei durch:

$$\triangle w_{ij}(t+1) = -\theta \cdot \frac{\partial E}{\partial w_{ij}} \tag{2.1}$$

In vielen Anwendungsfällen gibt es eine große Trainingsmenge an Daten. Anstatt den Fehler über die Gesamte Datenmenge zu berechnen werden kleine Stapel (*Batches*) gebildet. Der Gradientenabstieg und Gewichtsupdates erfolgen für diese Teilmengen. Im Extremfall besteht der *Batch* aus einer einzelnen Eingabe, was als on-line Gradientenabstieg oder stochastischer Gradientabstieg (SGD) bezeichnet wird.

Verbesserte Backpropagationverfahren

Trotz vieler Vorteile ist einer Nachteile von Backpropagation, dass das Verfahren lediglich in lokalen Umgebungen sucht. Probleme die dabei auftreten können sind:

- Lokale Minima
- Flache Plateaus mit nur sehr geringer Steigung, die dafür sorgen, dass der Gradientenabstieg stagniert
- Gute Minima werden durch zu hohe Lernraten übersprungen
- Oszillation an steilen Schluchten in der Fehlerebene

Momentum, Weight-Decay und Drop-Out

In den meisten Implementation von Backpropagation findet man diverse Zusätze, die den Problem entgegen wirken sollen.
Rumelhart und Hinton [Rumelhart et al., 1986] führten einen Momentum-Term $\alpha \triangle w(t)$ ein, welcher die bereits vollzogene Gewichtsänderung zum Zeitpunkt t bei der Berechnung der Änderung zum Zeitpunkt $t+1$ berücksichtigt.

$$\triangle w_{ij}(t+1) = -\theta \cdot \frac{\partial E}{\partial w_{ij}} + \alpha \triangle w_{ij}(t)$$

Dies erhöht die Gewichtsänderung in weiten Plateaus und bremst den Gradienten in zerklüfteten Fehlerflächen ab.

Zusätzlich wird versucht die Gewichte zu normalisieren. Große Gewichte, welche die Fehlerfläche zerklüften und die Wahrscheinlichkeit von Oszillation zunehmen lassen, sollen vermieden werden. Dies geschieht durch hinzufügen eines *Weight-Decays* $-\lambda \cdot w_{ij}(t)$.

$$\triangle w_{ij}(t+1) = -\theta \cdot \frac{\partial E}{\partial w_{ij}} - \lambda \cdot w_{ij}(t)$$

Neben dieser Regulierungsmethode kommt in den meisten Convolutional Neural Networks zusätzlich *Drop-Out* [Srivastava et al., 2014] zum Einsatz. Dabei werden in der Trainingsphase bestimmte Neurone mit einer vorher festgelegten Wahrscheinlichkeit deaktiviert. Sie werden für nachfolgenden Berechnungen nicht mit eingebunden. Vorteile die dabei entstehen sind bessere Generalisierungsleistungen der Netzwerke und der abnehmender Einflusses der Gewichtsinitialisierung. Zwei wichtige Faktoren die uns beim Trainieren von Convolutional Neural Networks begegnen.

2.2 Convolutional Neural Networks

Convolutional Neural Networks enthalten die selben grundlegenden Elemente eines vorwärts gerichtete künstliche Neuronale Netzes.
Ihre Eingaben besteht aus einem Bild, welches durch seine Pixelwerte als Vektor dargestellt werden kann. Dieser Bildvektor wird durch die Breite und Höhe des Bildes aufgespannt. Eine mögliche dritte Dimension bilden dabei die RGB-Daten jedes einzelnen Pixels. Jedes Bild kann durch seine Pixelwerte als Punkt im Raum dargestellt werden. Jeder Pixel bildet dabei eine eigene Dimension. Die Idee hinter der Klassifizierung ist es den Raum in dem alle Bilder liegen mittels Klassengrenzen zu separieren.
Die Ausgabe der Netze bildet eine Score Funktion, welche die Eingaben auf die entsprechenden Klassen abbildet. Ihre Werte werden als Wahrscheinlichkeiten

dafür angesehen, dass das Bild der entsprechenden Klasse zuzuordnen ist.
Der Ansatz der Convolutional Neuronal Networks ist es, Bildmerkmale zu extrahieren und die Bilddimensionen zu verkleinern. Das transformierte Bild wird nun nicht mehr anhand seiner Pixel sondern anhand seiner Bildmerkmale im entsprechenden, verkleinerten Raum dargestellt und separiert. Um diese Teilung zu lernen, müssen die Netzwerke zunächst trainiert werden. Durch eine Menge an Trainingsdaten soll das Netz Klassenmerkmale lernen, auf deren Basis es neue Daten klassifizieren kann. Es wird versucht ein Netzwerk so zu trainieren, dass es lernt anhand von Bildmerkmalen zu generalisieren. Dem gegenüber steht das Problem des *Overfittings* welches beim Training auftreten kann. Anstatt generalisierbare Klassenmerkmale zu lernen, spezialisiert sich das das Netzwerk auf die Trainingsbilder. Dadurch kann es nur diese richtig klassifizieren. Anzeichen für ein Overfitting sind Klassifizierungsleistungen für die Trainingsdaten nahe der 100% .

2.2.1 Aufbau und Architektur

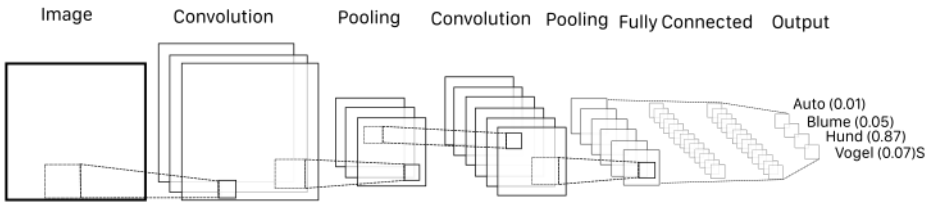


Abbildung 2.2: Aufbau eines klassischen Convolutional Neural Network (CNN)

Convolutional Neural Networks (Abbildung 2.2) setzen sich hauptsächlich aus drei Layer Arten zusammen : *Convolutiona Layer*, *Pooling Layer* und *Fully-Connected Layer*.

Convolutional Layer

Das Convolutional Layer erhält seinen Namen aufgrund seiner Faltungsoperation (Convolution), welche von einer reihe an Filtern (auch *Kernel* genannte) durchgeführt wird.
Ein Filter stellt dabei eine dreidimensionales Matrix dar, die sich über das Bild bewegt. In jedem Schritt wird das innere Produkt mit dem aktuell betrachteten Bildausschnitt (auch als rezeptives Feld bezeichnet) über die gesamte Tiefe berechnet. Dieser Prozess heißt Faltung oder Convolution. Das innere Produkt stellt dabei die Eingabe für ein Neuron im Convolutional-Layer dar.

Die Filterbreite und -höhe können variieren, während die Tiefe immer der der Eingabe entspricht. Die Einträge innerhalb der Faltungsmatrix sind Gewichte, die mit Hilfe eines überwachten Lernverfahrens angepasst werden. Anstatt einzelne Filter für jedes einzelne Neuron zu lernen, werden die Gewichte der Filter untereinander geteilt. Dadurch ist es möglich mit einer Faltungsmatrix über das gesamte Bild zu wandern und eine Faltung durchzuführen.
Die Ausgaben, die dabei entstehen, bilden eine zweidimensionale Matrix, auch *Feature-Map* genannt. Sie enthält die Filterausgaben für jede lokale Position. Da es sich bei der Faltung um eine rein lineare Operation handelt wird für die Neurone in den Feature-Maps eine nicht-lineare Aktivierungsfunktion benutzt. Üblicherweise eine *Rectified Linear Unit*, kurz (*ReLu*). Mehrere Feature-Maps ergeben das Convolutional Layer, wobei die tiefe abhängig von der Anzahl an Filtern ist.
Wichtige Eigenschaften der Convolutional Layer sind die lokale Konnektivität und die geteilten Gewichte. Sie sorgen dafür, dass Neurone durch bestimmte Bildeigenschaften (*Features*) unabhängig von deren lokalen Positionen aktiviert werden. Dadurch entsteht eine Translationsinvarianz.

Pooling Layer

Auf ein oder mehrere Convolutional Layer folgen Pooling Layer. In ihnen werden Informationen verworfen und die Dimensionen der Feature-Maps reduziert. Dahinter steckt die Annahme,dass die exakte Position eines Merkmals für die Objekterkennung nicht notwendig ist. Die meist genutzte Form des Poolings ist das sogenannte *Max-Pooling*. Für jede Feature-Map wird dabei das Maximum innerhalb eines kleinen quadratischen Fensters bestimmt, welches nach und nach über die Gesamte Höhe und Breite wandert. Durch die massive Datenreduktion entstehen Vorteile beim Rechenaufwand des Netzes, da weniger Parameter gelernt werden müssen. Ebenso wird die Invarianz gegenüber Transformationen, Translationen oder Rauschen im Bild verstärkt. Die rezeptiven Felder vergrößern sich in tieferen Schichten und das Overfitting Problem kann besser kontrolliert werden. Biologisch lässt sich das Pooling in gewisser Weise als laterale Hemmung verstehen.

Fully-Connected Layer

Durch Stapel an Convolutional- und Pooling-Layer extrahiert das Netz *low-' bis high-level' Features*. Darunter fallen Ecken und Kanten bis hin zu komplexeren Bildmerkmalen wie Gesichter. Diese Features werden mit Hilfe ein oder mehrerer Fully-Connected Layer verknüpft , um Ausgaben zu produzieren. Das Fully-Connected Layer (FC-Layer) kann dabei als klassischess Multi-Layer Perceptron 2.1.3 gesehen werden. Seine Neuronen sind vollständig mit

der Ausgabeschicht verbunden. Dabei korrespondieren die Neuronen der Ausgabeschicht mit den zu bestimmenden Klassen. Am Ende der Berechnungen wird die Ausgabe durch eine *Softmax-Funktion* in einen Bereich zwischen 0 und 1 gebracht, wobei sich die Summe zu 1 addiert. Der Ausgabewert eines output Neurons, wird als Wahrscheinlichkeit oder Vertrauensgrad des Netzwerkes für die entsprechende Klasse angesehen.

2.2.2 Training

Das Training der Gewichte erfolgt, wie schon bei klassischen Neuronalen Netzen, durch Backpropagation2.1.4. Im ersten Schritt werden alle Gewichte mit zufälligen, kleinen Werten initialisiert. Darauf wird eine Eingabe durch das gesamte Netz propagiert. Die Ausgabe des Netzes wird mit dem Bildlabel, der sogenannten *Ground Truth*, verglichen. Der dritte Schritt ist die Berechnung des Fehlers, auf dessen Grundlage die Gewichte angepasst werden.

2.2.3 Bekannte Architekturen

Während das Konzept der Convolutional Neural bereits seit den frühen 90er Jahren existiert, gewinnt es über die letzen Jahre immer weiter an Bedeutung. Wettbewerbe wie die ImageNet Large Scale Visual Recognition Challange (*ILSVRC*) [Russakovsky et al., 2012], die Visual Object Classes Challenge (*VOC*) oder das *ImageCLEF* Forum konnten von Convolutional Neural Networks gewonnen werden. Wettbewerbe , die als als Maßstab im Bereich der Bildverarbeitung dienen und immer wieder neue Ansätze hervorbringen. Auf den in 2.2.1 erläuterten Grundprinzipien aufbauend, entwickelten sich Architekturen, die für enorme Fortschritte in den letzten fünf Jahren sorgten. Beginnend mit Yann LeCuns *LeNet* [Lecun et al., 1998] und Alex Krizhevskys *AlexNet* [Krizhevsky et al., 2012] konnte ein neuer State of the Art gesetzt werden.

Kapitel 3

Teil 1. Replikation

3.1 PlantClef

PlantClef ist ein auf Pflanzen spezialisierter Bereich innerhalb des *ImageClef* Forums.
Als Teil des *Conference and Labs of the Evaluation Forum (CLEF)*, geht es bei ImageClef um die Förderung und Entwicklung von Systemen zum Informationsaustausch. Fokussiert werden dabei Fortschritte im Bereich visueller Medien Analyse, Indizierung und Klassifizierung. ImageClef biete seit 2003 die Ressourcen und die Infrastruktur zur Evaluation solcher Systeme.
Von [Clough and Sanderson, 2003] an begonnen, haben sich mittlerweile mehrere spezialisierte Untergruppen gebildet. Darunter auch PlantClef. Im rahmen diese Forums werden jährlich Wettbewerbe zur Pflanzen Erkennung veranstaltet. Ziel ist es, die Klassifizierung von Bilddatenbanken zu verbessern, welche insbesondere bei Aktuellen Problemen, wie der mechanischen Behandlung oder Wiedereinführung von Arten Anwendungen finden würde.

3.1.1 Aufgabe

Die Aufgabe der PlantClef Challange 2016 bestand darin, invasive Pflanzenspezies innerhalb einer durch crowdsourcing gesammelten Bilddatenbank besser aufzuspüren.
Für die Bilddatenbank bediente man sich der mobilen App *Pl@ntNet* [1], in welcher mehr als hunderttausend Nutzer täglich mehrere Tausend Pflanzenfotos hochladen und bestimmen. Jedes Bild wurde mit einem entsprechenden Label versehen, welches entweder bereits durch die User bestand oder mit Hilfe einer Reihe an Botanikern bestimmt wurde. Der endgültige Test Datensatz wurde aus 8000 Bilder zusammengestellt. 4633 davon enthielten bereits im Vorjahr verwendete Pflanzen, aus insgesamt 1000 Klassen. Die restlichen 3367 wurden

[1] http://identify.plantnet-project.org

mit 144 noch unbekannten Klassen wie Pilzen, Gemüse oder auch Tieren gelabelt. Unter den Pflanzenbildern gehörten 366 zu 26 potentiell invasiven Spezies. Die Liste an potentiell invasiven Arten wurden den Teilnehmern vorenthalten.
Die Aufgabe der Netzwerke war es nun, für jedes Bild die entsprechende Klasse zu finden und mit einem Wert zwischen 1 und 0 zu versehen (2.2.1). Bilder einer unbekannten Klasse, sollten ignoriert werden und nicht unter den Ergebnissen auftauchen.
Ein sogenanntes *open-world/open-set recognition Probleme* [Scheirer et al., 2014, Bendale and Boult, 2015]. Dabei kommt es darauf an, falsch-positive Fehler des Netzwerkes für unbekannte Objekte zu vermeiden. Entstehen können dieses dadurch, dass das Netzwerk jedem Objekt ein Label der von ihnen gelernten Klassen zuordnet.
Die Leistung der Netzwerke wurde durch die *Mean Average Precision (MAP)* bemessen. Jedes Team hatte die Möglichkeit bis zu vier verschiedene Testläufe einzureichen.

Mean Average Precision (MAP)

Für jede Klasse wird eine Liste geführt. Trifft ein Netzwerk eine Vorhersage für die entsprechende Klasse, wird das Bild samt Wahrscheinlichkeit absteigend in die Liste einsortiert. Für jede Klasse C_i wird die durchschnittliche Präzision (Average Precision) ermittelt:

$$AveP = \sum_{k=1}^{n} P(k) \triangle r(k)$$

Dabei gibt k den Rang der Liste an. $P(k)$ beschreibt die Präzision bis zum Rang k und $\triangle r(k)$ die Veränderung des Recalls von Rang $k-1$ zu k. Der Recall ist in diesem Fall definiert durch:

$$\frac{\left|\text{Bilder } \in \text{ C}_i \cap \text{ richtig Vorhergesagte Bilder}\right|}{\left|\text{Bilder } \in \text{ C}_i\right|}$$

Abschließend wird die Präzisionen über alle Klassen gemittelt. Fälle in denen unbekannte Objekte fälschlicherweise Klassifiziert werden, senken die Metrik drastisch. Hinsichtlich der 26 invasiven Arten führte man eine zweite MAP ein, wobei man in den Berechnungen nur diese Klassen berücksichtigte.

3.1.2 Offizielle Ergebnisse

Insgesamt acht Teams nahmen am Wettbewerb teil. Die besten unter ihnen erreichten sowohl für den gesamten Datensatz, als auch für eine Teilmenge aus invasiven Arten einen MAP von über 70%. Gewinner war das *Bluefield Netzwerk* [Hang et al., 2016] mit 74.2% bzw. 71.7%. Knapp dahinter mit einer Präzision von 73.8% bzw. 70.4% lag das *Sabanci System* von Mostafa Mehdipour Ghazi, Berrin Yankoğlu et al. [Ghazi et al., 2016], welches wir im weiteren Verlauf näher betrachten werden. Ebenfalls über die 70% Marke schaffte es das *CMP Netzwerk* [Śulc et al., 2016] mit 71% bzw. 65.3%.

3.2 Sabanci System

Das Team um Mostafa Mehdipour Ghazi [Ghazi et al., 2016] nutze für die Open-set recognition Aufgabe ein Ensemble aus mehreren bekannten CNNs. Das System beinhaltete das *Vgg16* [Simonyan and Zisserman, 2014] und *GoogLeNet* ([Szegedy et al., 2014]). Wobei es sich beim GoogLeNet um das *IncpetionV1* handelt. Beide Netzwerke waren auf dem ILSVRC 2014 [Russakovsky et al., 2012] Datensatz vortrainiert und wurden für die PlantClef Aufgabe neu abgestimmt. Für die Klassifizierung wurden die Ausgaben beider Netze addiert und gemittelt.
Neben den zwei, auf Pflanzen Klassifizierung trainierten Netzen, kam ein drittes Netzwerk für die Unterscheidung von Pflanzen und nicht pflanzlicher Objekten zum Einsatz. Hierfür verwendete das Team um Ghazi ein weiteres GoogLeNet, welches sie auf einem separaten Datensatz trainierten. Die Idee war dadurch schon frühzeitig nicht pflanzliche Objekte und unbekannte Klassen auszusortieren.

3.2.1 VGG16

Bei dem *Vgg16* handelt es sich um ein tiefes neuronales Netz, was 2014 von Koren Simonyan und Andrew Zisserman [Simonyan and Zisserman, 2014] entwickelt wurde. Bekanntheit erlangte es durch die Teilnahme an der ILSVRC 2014. Mit einem *Top-5-Fehler* von 7.3% gewannen Simonyan und Zisserman den zweiten Platz und konnten den Vorjahres-Gewinner [Sermanet et al., 2013] um knapp 4% unterbieten. Bei einem *Top-N-Fehler* wird geprüft, ob die Klassen mit den N größten Werten das richtige Label enthalten.
Hauptaugenmerk legten Simonyan und Zisserman auf die Tiefe ihres Netzes, welche wie sie zeigen konnten einen großen Einfluss auf die Leistung hat. Die durch Rechenleistung limitierte Tiefe vergrößerten die beiden durch den Einsatz von kleineren hintereinander gestapelten Filtern. mit Hilfe von 3×3

und 1×1 Filtern erreichte ihre Architektur eine Tiefe von 16, bis hin zu 19 gewichteten Layern.
Basierend auf den Prinzipien von Ciresan [Ciresan et al., 2011] und Krizhevsky [Krizhevsky et al., 2012], benutzt das VGG16 eine Reihe an kleinen 3×3 Filtern, gerade Groß genug um Unterscheidungen von links und rechts, oben und unten oder zentral zu treffen. Vorteile liegen im Vergleich zur bisher verwendeten $7x7$ Filtern [Zeiler and Fergus, 2013] in der Anzahl an Parametern. Ein $7x7$ Filtern besitzt ein effektives rezeptives Feld der Größe $7x7$, was ebenfalls durch das hintereinander Schalten von drei 3×3 Filtern erreicht werden kann. Während $7x7$ Filter bei einem Input und Output mit C Kanälen $7^2 \cdot C^2 = 49 \cdot C^2$ Gewichte benötigen, sind es bei einem dreier Stapel an 3×3 Filtern lediglich $3 \cdot (3^2 \cdot C^2) = 27 \cdot C^2$. Zusätzlich bietet der Stapel kleinerer Filter die Möglichkeit weitere, nicht-lineare Aktivierungsfunktionen zwischen den Filtern einzusetzen und so die Entscheidungsfunktion zu verbessern. Ähnliches gilt auch für den Gebrauch von 1×1 Filtern, welche keinen Einfluss auf die effektiven rezeptiven Felder haben aber die non-Linearität der Entscheidungsfunktion erhöhen. Simonyan und Zisserman verwendeten sie zur lineare Transformation der Input Kanäle, auf die eine non-lineare Aktivierungsfunktion folgte.

Tests für verschiedene Konfigurationen des Netzes zeigten, dass sowohl das Erhöhen der Tiefe als auch die zusätzliche Non-Linearität die Klassifizierungsleistung verbesserten. Simonyan und Zisserman erkannten zwar, dass 1×1 Filter bei der Klassifizierung halfen, jedoch schien es den Ergebnissen nach wichtiger zu sein, die Räumliche Auflösung beizubehalten. Sie kamen in der endgültigen Architektur nicht vor.
Im direkten Vergleich zwischen 5×5 und 3×3 Filtern konnte das Netz mit einer Reihe an 3×3 Filtern die besten Ergebnisse erzielen. Die Fehlerrate in ihrer endgültigen Architektur stagnierte nach einer Tiefe von 19 Layern.
Abbildung 3.1 zeigt die finale Architektur des 16 Layer Netzwerkes: VGG16.

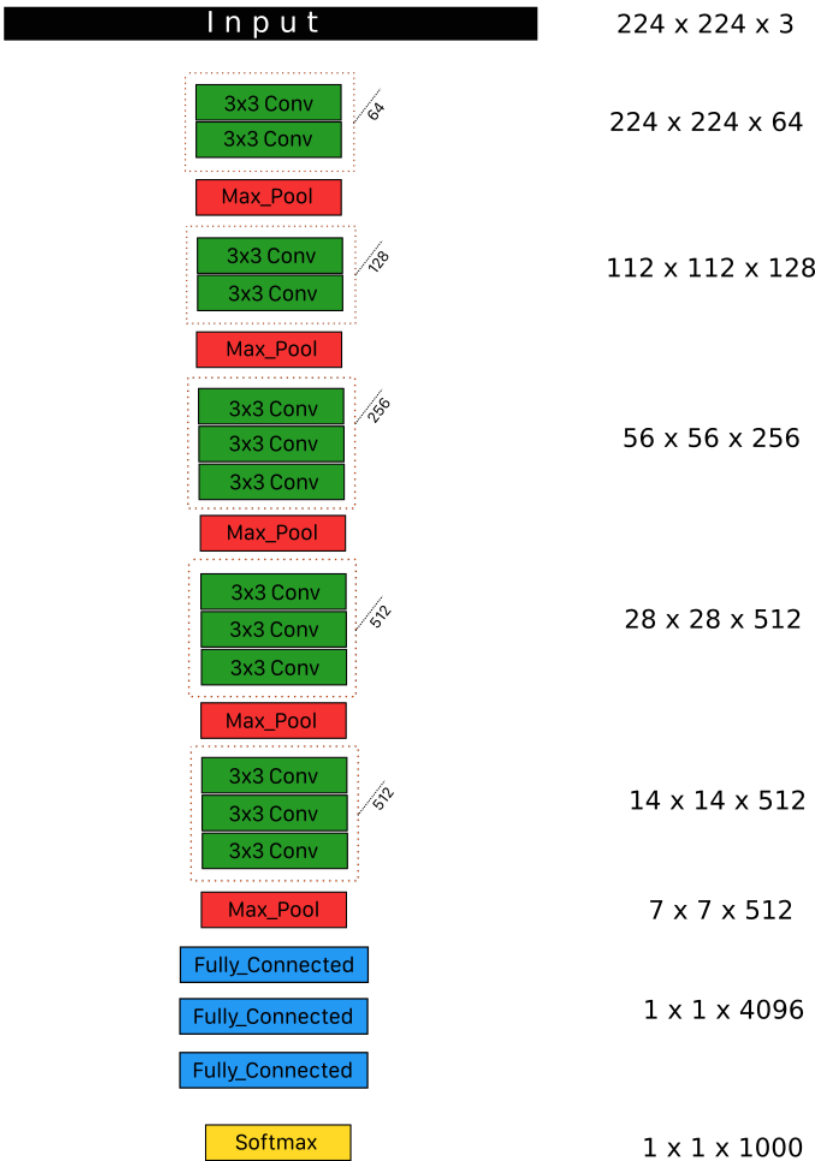


Abbildung 3.1: Die Architektur des VGG16 von Koren Simonyan und Andrew Zisserman. Charakteristisch sind die Stapel Convolutional Layer mit kleinen 3×3 Filtern. Diese benötigen weniger Rechenaufwand und ermöglichen tiefere Netzwerkstrukturen.

3.2.2 GoogLeNet

2014 entwickelten Christian Szegedy, Wei Liu et al. [Szegedy et al., 2014] das sogenannte *GoogleNet*, welches im selben Jahr große Bekanntheit erlangte. In der ILSVRC 2014 konnte es einen *Top-5-Error* von 6.67% erreichen und setzte sich gegen das VGG16 von Simonyan und Zisserman durch. Unabhängig von einander erkannten beiden, dass die Tiefe eines Netzes eine Rolle zu spielen scheint.
Der Name Inception, der häufig im Zusammenhang mit dem GoogleNet fällt, geht auf die Module zurück aus denen das GoogleNet aufgebaut ist. Sie stellen eine eigene Netzwerkarchitektur dar und bilden damit Netzwerke im Netzwerk. Insgesamt neun hintereinander gestapelt Module sind es aus denen das GoogLeNet besteht.
Inspiriert durch Serres et al. [Serre et al., 2007] Modell des primären visuellen Kortex , hatten Szegedy et al. die Idee eine Serie an unterschiedlich große Filter zu benutzen, um so unterschiedlich große Bildeigenschaften zu detektieren. Neben der Tiefe vergrößerte dieser Ansatz zusätzlich die Breite des Netzwerkes. Anstelle eines einzelnen Convolutional, Pooling oder FC layer trat eine Gruppe an Layern. Beides bedeutete mehr Gewichte, die es zu lernen galt und einen enormen Anstieg in der benötigten Rechenleistung. Vor dieses Problem gestellt, entwickelten Szegedy et al. die so genannten *Inception Module* (Abbildung 3.3).
Die Hauptidee hinter den Incpetion Modulen beinhaltet die lokalen Strukturen des Netzwerkes spärlich zu halten, sprich die lokale räumliche Auflösung vorheriger Layer möglichst kompakt zu repräsentieren. Einen Ansatz den Arora [Arora et al., 2013] 2013 präsentierten, bestand darin die Bereiche eines Layers, welche eine hoher Korrelation besitzen, in Clustern zusammenzufassen. In den ersten Ebenen fassen diese Cluster einzelne Bildbereiche zusammen und können somit durch 1×1 Filter im nächsten Layer abgedeckt werden. Für räumlich ausgedehntere Bereiche, die über mehrere Cluster verteilt sind, wählt man größere Filter. Im Inception Module kommen dafür neben den 1×1 Filtern 3×3 und 5×5 Filter zum Einsatz.
Die zweite Idee war es, die aufwendigen 5×5 und 3×3 Filter Berechnungen effizienter zu gestalten. Dazu reduzierte man die Dimensionen des Eingabe Layers mit Hilfe von 1×1 Filtern. Das Eingabesignal wurde komprimiert. Im *InceptionV1* wurden die 5×5 Filter durch weiter 3×3 Filter ersetzt. Neben den Convolutional-Layern wurde ein paralleles Pooling-Layer eingesetz. Die Ausgaben aller Operationen fasste man zu einem Ausgabevektor zusammen.

Ein Problem unter dem auch das GoogleNet leidet, ist das des *Vanishing Gradients*. Das Fehlersignal, welches mit Hilfe von Backpropagation durch das Netz getragen wird, erreicht frühe Schichten im Netzwerk nur sehr abgeschwächt oder gar nicht. Anpassungen der Gewichte sind daher kaum möglich.

Um dieses Problem abzumildern, schalteten Szegedy et al. während des Trainings weitere Ausgabeschichten mit Softmaxclassifier zwischen. Zum eintreffende Fehlersignale werden die zusätzlichen Fehlersignale addiert und somit das ursprüngliche Signal verstärkt. (Abbildung **??**

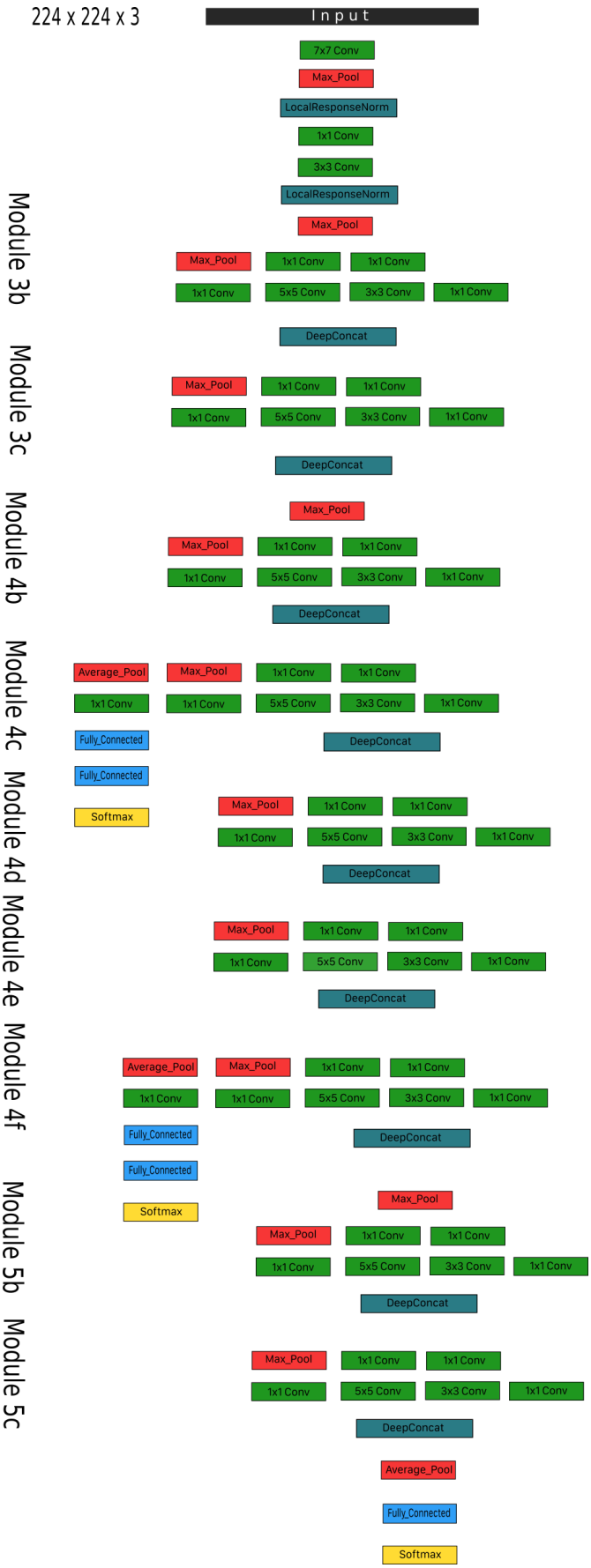


Abbildung 3.2: Die Architektur des GoogLeNet von Christian Szegedy. Auch bekannt als InceptionV1. Die Besonderheit stellen die Inception Module dar. 1×1 ermöglichen die parallele Anwendung von Faltungs- und Poolingoperationen

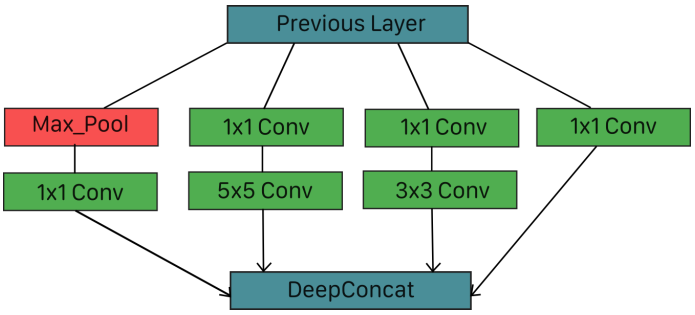


Abbildung 3.3: Ein einzelnes Inception Module, bestehend aus mehreren Convolution Operationen und parallelem Max-Pooling. Allen Operationen sind 1×1-Filter vorgeschaltet. Diese dienen der Korrelationsberechnung zwischen den einzelnen Eingabekanälen und reduzieren so die Dimensionen. Alle Feature-Maps laufen zusammen und bilden gestapelt die nachfolgende Ebene.

Neben der Größe eines Netzes und seiner Architektur sind es die Daten die ausschlaggebend dafür sind, wie gut das jeweilige neuronale Netz lernen kann. Je mehr Daten zur Verfügung stehen, desto mehr Trainingsiterationen sind möglich. Zusätzliche Trainingsiterationen ermöglichen zusätzliche Gewicht Updates, welche das Netzwerk besser und robuster auf seine Aufgabe abstimmen. Aus diesem Grund sind Datenvergrößerung (*Data Augmentation*), Daten Vorverarbeitung (*Preprocessing*) und die Einteilung der verfügbaren Daten (*Split*) große Themen bei der Entwicklung guter Neuronaler Systeme. Erste Schritte beim Umgang mit Daten sind meist die Vorverarbeitung und Vergrößerung der Datenmenge. Um einen bestehenden Datensatz zu vergrößern, werden einfache Transformationen der Daten vorgenommen und so neue Bilder generiert. Bestehende Bilder werden rotiert, skaliert, in bestimmten Bildbereiche ausgeschnitten oder verrauscht. Ebenso ist es gängig die Daten durch Abziehen des Bildmittelwertes zu zentrieren.
Ist das Bearbeiten der Daten abgeschlossen, kann die Trainingsphase beginnen. Um den Trainingsfortschritt zu überwachen und mögliches Overfitting zu erkennen, teilt man den Datensatz auf. Während man auf Teilen das Netzwerk trainiert, werden anderen Teile für die Validierung benutzt. Für sie muss das Netzwerk Vorhersagen treffen, die Aufschluss über den Verlauf des Trainings geben.

3.3 Der Datensatz

Im Rahmen der Plantclef challenge 2016 stand eine Sammlung an 113205 Bildern zur Verfügung. Die Bilder umfassten Bäume, Kräuter und Farne aus dem französischen und direkt angrenzenden Pflanzenreich. Insgesamt ließen sich 1000 verschiedene Spezies unterscheiden.
Zu jedem Bild existierte eine XML Datei,die alle relevante Informationen über das Bild enthielt. Neben der genauen Taxonomie enthielt diese den Ort an dem das Bild entstand, den Zeitpunkt und auch den zusehenden Abschnitt der Pflanze. Im Bild enthalten konnten dabei der Ast (*branch*), die gesamte Pflanze (*entire*), die Blüte (*flower*), die Frucht (*fruit*), das Blatt (*leaf*), der Stamm (*stem*) oder eine Großaufnahme eines Blattes vor weißem Hintergrund (*leafScan*) sein. Als Label der Bilder wurde eine mitgelieferte *ClassId* benutzt. Eine numerisch Zahl die zur taxonomischen Bestimmung im französischen Botaniker Netzwerk, Tela Botanica, verwendet wird.

3.3.1 Einteilung des Datensatzes

Die Insgesamt 113205 Bilder setzen sich aus einem Trainings- und Testdatensatz zusammen, welche beide für den Wettbewerb im Jahr zuvor verwendet wurden. Der Trainingsdatensatz enthielt 91758 Bilder, während die restlichen 21446 erst nach ende des Wettbewerbs veröffentlicht wurden. 2016 standen beide von vornherein zur Verfügung.
Zum Trainieren der Netze teilten Ghazi et. al den Trainingsdatensatz von 2015 zufällig. 70904 Bilder verwendeten sie für das Trainieren der Netze, 20854 waren ausschließlich für die Validierung vorgesehen.
Der Testdatensatz von 2015 kam ebenfalls für Trainingszwecke zum Einsatz, indem aus allen zur Verfügung stehenden Bildern ein großer Trainingsdatensatz mit 102777 Bilder (Trainigsdatensatz + Testdatensatz + $\frac{1}{2}$ Validierungsdatensatz) erstellt wurde.
Als zusätzliche Daten verwendete das Team knapp 90000 Bilder nicht pflanzlicher Objekte aus der ILSVRC 2012, welche gepaart mit den 91758 Pflanzen Trainingsbildern einen knapp 180000 Bilder großen Datensatz ergaben. Benutzt wurde dieser, um ein Netzwerk auf die binäre Entscheidung zwischen Pflanze oder Nicht-Pflanze zu trainieren. Wie zuvor wurden diese Bilder in Training und Validierung aufgeteilt. Zu den bereits vorhandenen 70904 Pflanzen Trainingsbildern fügte man ungefähr gleich viele Nicht-Pflanzen hinzu, welche Zusammen den Trainingsdatensatz von knapp 140000 bildeten. Analoges tat man für den Validierungsdatenssatz.

Alle Daten können auf den Internetseite *imageclef.org* [2] und *image-net.org*

[2] http://www.imageclef.org/lifeclef/2016/plant

Datensätze			
Name	Training	Validierung	Zusammensetzung
Training2015	70904	20854	Trainingsdaten 2015
Gesamt2015	102777	10427	Trainings- und Testdaten 2015
Binär2015	ca.140000	ca.40000	Trainingsdaten + ILSVRC 2012
Test2016	8000		Testdaten 2016

Tabelle 3.1: Die verwendeten Datensätze im Überblick.

[3]herunter geladen werden.

3.3.2 Data Augmentation und Vorverarbeitung

Neben dem Spalten der Datensätze, war die Vorverarbeitung und Augmentierung der Daten ein großer Bestandteil der Arbeit von Ghazi et al.. Mit Hilfe von Ausschneiden, Skalieren und Spiegeln vervielfachte man jedes einzelne Bild. Dazu ging man wie folgt vor:
Zunächst wurden K Ausschnitte aus der Bildmitte entnommen. Das Original Bild rotierte man um $\pm R°$ und schnitt aus jeder rotierten Version die Bildmitte aus. Alle Ausschnitte, sowie das Original Bild skalierte man auf 256×256 Pixel. Um die Bilddaten zu zentrieren, zog man in allen RGB-Kanälen den Mittelwert ab. Zum Schluss wurde jedes der $K + R + 1$ Bilder an den vier Ecken und in der Bildmitte ausgeschnitten. Jeden Ausschnitt spiegelte man horizontal, was jedes einzelne Bild um $10 \cdot (K + R + 1)$ vervielfachte. In unserer Arbeit ließen wir das Spiegeln der Ausschnitte aus. Grund dafür war die Vermutung, dass durch eine horizontale Spiegelung wichtige Pflanzenmerkmale, wie die Anordnung der Blütenblätter, verloren gehen könnten.

3.4 Hardware

Die Technische Realisierung unserer Arbeit erfolgte auf einem Ubuntu System mit Intel(R) Core(TM) i5-3470 Prozessor. Dieser besaß eine Grundtaktfrequenz von 3.2 Ghz und war mit einer NVIDIA GeForce GTX 1060 gekoppelt, welche 6GB besaß.

3.5 Methodik und Implementierung

Für die Implementierung der Arbeit verwendeten wir Python im Zusammenspiel mit *TensorFlow1.2* und der darauf aufbauenden Bibliothek *TensorFlow-Slim*. Bei TensorFlow handelt es sich um eine open-source Software Bibliothek, die vom Google Brain Team entwickelt wurde. Eingesetzt wurde diese

[3]http://image-net.org/challenges/LSVRC/2012/browse-synsets

ursprünglich in der internen Forschung im Bereich des maschinellen Lernens und tiefer neuronaler Netze. Seit 2015 ist sie öffentlich verfügbar.
Die Funktionsweise von TensorFlow lässt sich in zwei Abschnitte unterteilen:

i Das Aufbauen eines Berechnungsgraphen (computational graph)

ii Das Ausführen der Berechnungsgraphen

Grundelement innerhalb des Graphens sind Tensoren. Bei einem Tensor handelt es sich um eine mathematische Funktion, welche die lineare Beziehung zwischen Vektoren, Skalaren oder anderen Tensoren beschreibt. Gegeben einer Basis an Vektoren, lässt sich ein Tensor numerisch als eine n-dimensionale Matrix darstellen. Berechnungen werden auf Tensoren durchgeführt. Jeder Tensor kann dabei entweder als Konstante mit bereits vordefinierten Werten bestehen, als Platzhalter für mögliche Eingaben dienen oder als Variable für zu trainierende Gewichte. Tensoren und Rechenoperationen werden in TensorFlow als Knotenpunkt dargestellt, welche miteinander verknüpft den Berechnungsgraphen aufbauen. Für die letztendliche Berechnung müssen innerhalb einer sogenannte TensorFlow *Session* alle Variablen initialisiert und alle Platzhalter mit Eingaben ersetzt werden. Tensorflow-Slim bietet dabei eine sehr große Hilfe, da hier diverse TensorFlow Funktionen bereits vordefiniert sind. Darüber hinaus besitzt Tensorflow-Slim eine Sammlung an bereits erstellten und vortrainierten Netzwerken, darunter auch das InceptionV1 und das VGG16, welche wir in unserer Arbeit verwendeten.

Trainingsphase

In der Replikation der Ergebnisse konzentrierten wir uns auf den besten Testlauf des Sabanci Systems [Ghazi et al., 2016].
Für die verwendeten Netze InceptionV1 und VGG16 nutzten wir Tensorflow-Slim Implementierungen. Beide mit bereits vortrainierten Gewichten. Die Daten konvertierten wir in das TensorFlow eigenen TFRecord format.
Zunächst trainierte das Team um Ghazi das InceptionV1 für 600.000 Iterationen auf dem Trainingsdatensatz mit einer Batchgröße von 20. Für jedes Bild setzten sie die Anzahl quadratischer Ausschnitte $K = 9$ und die Rotationsgrade auf $R = \{10, 20\}$. Da wir keine Spiegelungen vornahmen resultierten daraus insgesamt 70 Ausschnitte pro Bild. Um Rechenaufwand zu sparen berechneten wir nicht alle 70 Ausschnitte, sondern wählten unter den K Ausschnitten, dem Original und den rotierten Varianten ($K + 1 + 2 * R = 14$) zufällig eines aus. Für dieses erstellten wir die Ecken- und Zentrumsausschnitte, unter denen wir erneut zufällig eins wählten. Diesen Vorgang nahmen wir für 20 zufällig aus dem Datensatz entnommenen Bilder vor, welche zusammen unseren Batch ergaben. Wir stellten dadurch eine gute Verteilung der Bilder innerhalb eines Batches sicher.
Ghazi et al. wählten die *Kreuzentropie* als Fehlerfunktion, die sie mittels Gradienten Abstiegs minimierten. Die Anpassung der Gewicht berechneten sie durch Backpropagation mit einer Lernrate von $\theta = 0.001$, die sie alle 12000 Iterationen exponentiell um 4% senkten. Zum Gewichtupdate addierten sie

ein Momentum-Term, den sie auf $0.9 \cdot \triangle w(t-1)$ setzten. Zu Regularisierungszwecken implementierte das Team ein Weight-Decay [Werbos, 1988] von 0.0002.
Alle 5000 Iterationen evaluierten wir die Performance auf dem Validierungsdatensatz, indem wir die *Top1* und *Top5 Genauigkeit* der Vorhersagen ausgeben ließen. Für die Berechnung der Vorhersagen, generierten wir alle $5 * (K + 1 + 2 * R)$ Bildausschnitte. Zusammen in einem Batch bildeten sie die Eingabe des Netzwerks. Die Ausgabewerte wurden über alle Ausschnitte gemittelt und die Klassen mit den maximalen Werten bestimmt. Diese vorhergesagten Klassen wurden mit dem Bildlabel verglichen und die Anzahl Übereinstimmungen berechnet.
Im Anschluss an das Training erweiterten Ghazi et al. den Trainingsdatensatz (siehe Tabelle 3.1) und führten weitere 200.000 Iterationen durch.
Das Vgg16 trainierten sie in gleicher Weise für 500.000 Iterationen. Im Gegensatz zu vorher resultierte die Vorverarbeitung der Daten in insgesamt 40 Ausschnitte pro Bild. Hierfür wurde $K = 5$ und die Rotationsgrade $R = \{10\}$ festgelegt.

Abschließend führte das Team eine Evaluation über dem gesamte Validierungsdatensatz durch. Dabei kombinierten sie die Ausgaben beider Netze. Um zu einer gemeinsamen Vorhersage zu kommen, bestimmten sie die Ausgabewerte der Netze für das entsprechende Bild und mittelten diese. Das System schaffte so ein Top1 Genauigkeit von 79.8%. Um unbekannte Objekte auszusortieren trainierten Ghazi drittes Netzwerk auf binäre Vorhersagen. Sie benutzen ein weiteres GoogleNet, welches mit den Gewichten des vorher auf Pflanzenerkennung abgestimmten Netzwerkes initialisiert wurde. Als Datensatz verwendeten sie die Trainingsbilder des PlantClef Datensatzes und eine gleiche Anzahl nicht-planzlicher Bilder. Für die nicht-pflanzlichen Objekte bedienten sie sich Daten der ILSVRC 2014 Challenge. Wir wählten aus den selben Daten zufällig nicht-pflanzliche Klassen aus, bis wir knapp 90.000 Bilder gesammelt hatten. Die Vorverarbeitung der Bilder resultierte in diesem Fall in 10 Ausschnitten pro Bild. Dabei wurde $K = 0$ gesetzt und keine weiteren Rotationen durchgefürt. Ghazi et al. schickten fr Ingesamt 100.000 Iterationen Batches aus 20 Bildern durch das Netz und passten wie zuvor die Gewichte an. Das Resultat waren knapp 100% Top1 Genauigkeit auf dem Validierungssatz.
Neben dem GoogleNet, welches auf die Unterscheidung pflanzlicher von nicht-pflanzlicher/unbekannter Objekte trainiert war, setzten Ghazi et al. einen Schwellenwert T ein, um unbekannte Klassen auszusortieren. Bilder, die das binäre Netzwerk als unbekannt klassifizierte und deren Vorhersagewert unter dem Schwellenwert lag, wurden verworfen. Den Schwellenwert bestimmten sie indem beide Netzwerke zur Pflanzenerkennung auf dem nicht-pflanzlichen Teil des binären Datensatzes getestet wurden. Die Resultate zeigten eine Gleichverteilung der Ausgabewerte, auf deren Grundlage sie einen Schwellenwert von 0.4 wählten. Der Schwellenwert galt für die maximalen Vorhersagewerte eines Netzwerkes.
Die offiziellen Test Ergebnisse der PlantClef Challange 2016, die Ghazi et al. durch ihr Ensemble erreichten, sie in Tabelle 3.3 aufgelistet.

Ergebnisse			
Netzwerk	Datensatz	Top-1	Top-5
InceptionV1	Training2015/Validierung	75.33	87.5
IncpetionV1	Gesamt2015/Validierung	80.14	91
Vgg16	Training2015/Validierung	73.43	85.93
Vgg16	Test2015	55.1	75.4
InceptionV1 + Vgg16	Test2015	70.5	87.2
InceptionV1-Binär	Binär2015	99.5	-

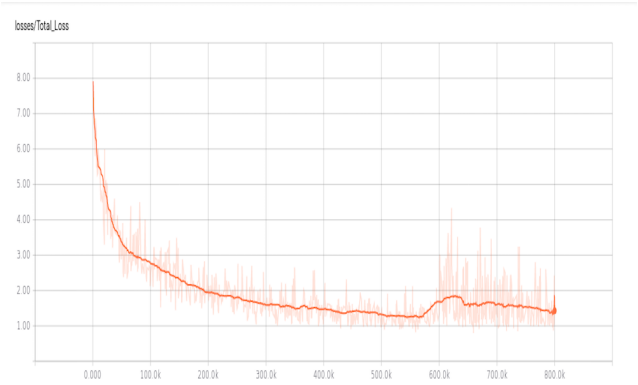
Tabelle 3.2: Die Top-1 und Top-5 Genauigkeiten der Netzwerke für Validierungsdaten im Überblick

Test Ergebnisse					
-	MAP closed-World	MAP invasiv	MAP open-World	T	# Verworfen
Offiziell	0.806	0.704	0.738	0.5	480
Replikation	0.727	0.61	0.675	0.5	1236+1434
Replikation	.77	0.641	0.707	0	1236

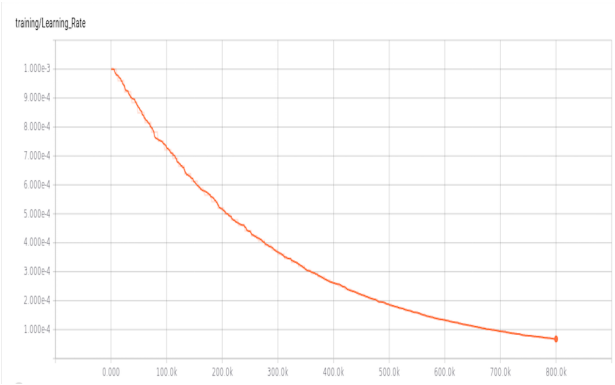
Tabelle 3.3: Ergebnisse des replizierten Sabanci Systems für den Testdatensatz des PlantClef Wettbewerbs 2016. Im Vergleich stehen die offiziell erzielten Resultate von Ghazi et al.

3.5.1 Ergebnisse

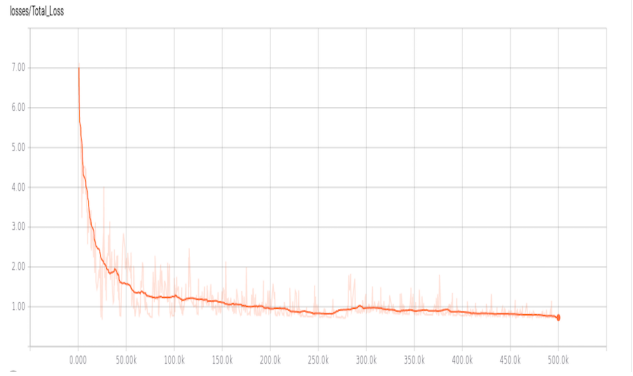
Der Verlauf unseres Trainings ist in den Abbildungen 3.4 zu sehen. Für unsere Pflanzenklassifizierer erreichten wir in unseren Evaluierungsdurchläufen eine Genauigkeit von 70.5% auf dem Testdatensatz. Unser binär trainiertes InceptionV1 konnte mit fast 100% Genauigkeit Pflanzen von nicht-pflanzlichen Objekten unterscheiden (Tabelle 3.2). Wir evaluierten das gesamte System auf dem Testsatz von 2016. Dieser umfasste sowohl nicht pflanzliche Objekte als auch Bilder die explizit als invasive Pflanzenspezies gelabelt waren. Für die Auswertung brachten wir die Vorhersagen unseres Systems in das von PlantClef gewünschte Format. Für jedes Bild gaben wir die entsprechende Bild-Nummer, sowie die vorhergesagte Klasse samt Wahrscheinlichkeitswert (¡ImageId; ClassId; Probability¿) an. Wir entschieden uns für jedes Bild die Wahrscheinlichkeiten für jede Klasse in unsere Testfiles aufzunehmen. Schlecht erkannte Klassen, die in den Evaluierungslisten weiter hinten standen, flossen dabei positiv in die Berechnung ein. Die Metrik selbst berechneten wir mit einem von PlantClef zur Verfügung gestellten Skript.
Wir erreichten eine *Mean Average Precision* von 67.5 % auf Testdatensatz. Potentiell invasive Spezies konnten mit einer *Mean Average Precision* von 61% erkannt werden. Dabei verwarfen wir 1236 Bilder als nicht-pflanzliche Objekte und 1434 aufgrund niedriger Ausgabewerte. Unsere Ergebnisse blieben unterhalb derer, die das Sabanci System 2016 im offiziellen PlantClef Wettbewerb erzielte (Tabelle 3.3).



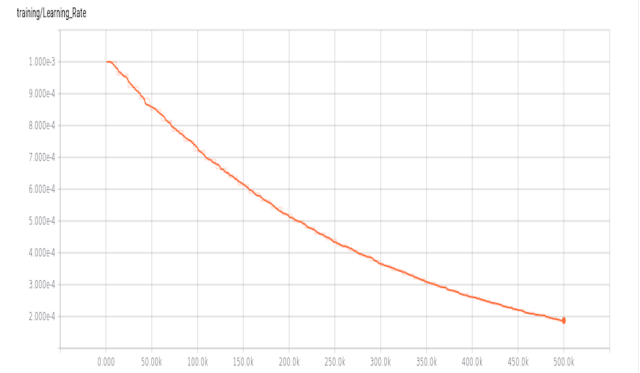
(a) Fehler: InceptionV1



(b) Lernrate: InceptionV1



(c) Fehler: Vgg16



(d) Lernrate: Vgg16

Abbildung 3.4: Die Abbildung zeigt die Entwicklung des Fehler bei zunehmende Trainingsiterationen. Die Lernrate passten wir dabei exponentiell an.

Kapitel 4

Teil 2. Analyse und Inspektion

Der zweite Teil unserer Bachelorarbeit befasste sich mit der Analyse und Inspektion der Netze samt ihrer Architekturen. Trotz beeindruckender Ergebnisse in Bereichen der Objekterkennung besteht immer noch kein klares Verständnis über die inneren Prozesse und das Verhalten Convolutional Neural Networks.
Matthew Zeiler und Rob Fergus erkannten bereits kurz nach dem beeindruckenden Erfolg des AlexNets , dass ein solches Verständnis wichtig für die Entwicklung guter Netzwerkarchitekturen ist. Fortschritte ließen sich andernfalls auf ein reines Trial-and-Error Spiel reduzieren. In ihren Arbeiten von 2014 [Zeiler and Fergus, 2013, Zeiler et al., 2011] entwickelten sie einen Ansatz, um die Aktivierung einzelner Filter auf das Eingabebild zurück zu projizieren. Bildeigenschaften, die den entsprechenden Filter aktivierten, konnten so visualisiert und potentielle Probleme des Netzes bei der Merkmalsextraktion aufgedeckt werden. Durch dieses Verfahren erkannten Zeiler und Fergus Schwächen, die in den ersten beiden Layern des AlexNets auftraten. Sie reduzierten die Filter Größe im ersten Layer von $11x11$ auf $7x7$ und erzielten dadurch bessere Ergebnisse.
In unserer Arbeit nutzten wir Zeiler und Furgus' Ansatz zur Visualisierung einzelner Filter über alle Layer hinweg. Durch die resultierenden Bilder versuchten wir Rückschlüsse auf Bildmerkmale zu ziehen, welche der Filter aktivierten und untersuchten, ob diese mit Bestimmungsmerkmalen korrespondierten. Außerdem platzierten wir Fully-Connected Layer an unterschiedlichen Ebenen der Netzwerke, um so die Vorhersagekraft für bestimmte Layertiefen zu ermitteln. Neben der Vorhersage auf Speziesebene, ließen wir die Netzwerke Vorhersagen für das Genus, die Familie oder das im Bild enthaltene Pflanzenorgan treffen. Im Zusammenspiel mit der Visualisierung prüften wir, ob die Netzwerke ein taxonomisches Klassifizierung bei der Pflanzenerkennung anwendeten und ob dieses durch die Netzwerktiefe beschrieben werden konnte.

4.1 Deconvolution

Die Technik, die Zeiler und Fergus anwendeten war ein sogenanntes *Deconvolution* Network [Zeiler et al., 2011]. Diese stellt eine Art Inverses zum Convolution Network dar. Dem trainierten Netz wird für jedes Layer ein entsprechendes Deconvolutional Layer hinzugefügt. Die Aktivität, die ein Eingabebild in den einzelnen Schichten hinterlässt, wird schrittweise durch Unpooling und inverses Filtern bis hin zur Eingabeschicht zurück geführt.

i) *Unpooling*: Pooling operationen , wie das oft verwendete Max-Pooling, sind Funktionen, die nicht invertierbar sind. Um dennoch eine Approximation zu erhalten, werden so genannte *Switch Variabeln* [Zeiler and Fergus, 2013] angelegt, welche die Positionen des Maximums enthalten. Mittels dieser Variablen ist es möglich, das rückwärts gerichtete Signal der entsprechenden Stelle zuzuordnen und so letztendlich die Stellen der maximalen Aktivität in der Eingabeschicht abzubilden.

ii) *Inverses Filtern*: Die Deconvolution Filter werden durch horizontales und vertikales Spiegeln des ursprünglichen Filters gebildet.

Angewendet werden alle Operationen auf der Aktivierungseingabe aus tieferen Schichten, die zuvor durch eine ReLU Funktion angepasst wurde.
An der Eingabeschicht angelangt stellt das rückwärts gerichtete Signal einen kleinen Teilbereich dar. Dieser Bereich enthält Bildmerkmale, die eine Maximale Aktivierung in dem untersuchten Filter auslösten.

4.2 Durchführung

Visualisierung

Für jede Schicht unserer Netzwerke berechneten wir die neun größten Aktivierungen jedes einzelnen Filters. Die Bilder, die wir dabei benutzen stammten aus dem Trainingsdatensatz. Wir erhofften uns für späteren Schichten komplexere Bildmerkmale zu sehen, welche auch als Klassifizierungsmerkmalen zur Bestimmung der Pflanzenspezies benutzt werden können. Abbildungen 4.2 - 4.15 zeigen zufällig gewählte Filter für die verschiedene Schichten der Netzwerke. Neben der Visualisierten Aktivierung sind die korrespondierenden Eingabebilder zu sehen.

Regression

Des weiteren untersuchten wir die Fragestellung, ob in der Tiefe der Netze eine taxonomische Klassifizierung erkennbar ist. Unser erster Versuch die Vorhersagekraft einzelner Schichten zu untersuchen bestand darin, eine Regressionsanalyse durchzuführen. Die Regressionsaufgabe war gestellt durch:

$$L = A \cdot B + \epsilon$$

L ist die Matrix der Label. Sie enthält für jedes Bild das zugehörige Label in One-Hot-Kodierung. Dabei besteht das Label aus Klassen korrespondierenden Bits. Der zur richtigen Klassen zugehörige Bit wird auf 1 gesetzt, der Rest bleibt 0. Bei N Bilder und K Klassen besitzt L die Dimensionen $N \times K$. A ist die Matrix der Erregungsmuster. Sie enthält für jedes Bild das Erregungsmuster aller Filter der jeweiligen Schicht. Die Dimensionalität ist $N \times J$. Um die dreidimensionalen Layer in einer Dimension darstellen zu können, werden die einzelnen Feature-Maps der Höhe nach aufgespalten und hintereinander gelegt. Bei einer Feature-Map der Größe 224×224 und einer Filter Anzahl von 64 ergibt sich $j = 224 \cdot 224 \cdot 64$. B ist die Matrix der Regressionskoeffizienten. Gelöst wird die Regressionsaufgabe durch:

$$B = (A^\top \cdot A)^{-1} \cdot A^\top \cdot S.$$

Wobei die Lösung der kleinsten Quadrate durch $A \cdot B$ gegeben ist und die Summe der quadratischen Abweichungen $||S - A \cdot B||$ als Fehlermaß genommen werden kann.

Die hohe Dimensionalität der Layer sorgten dafür, dass dieser Ansatz an fehlender Rechenleistung scheiterte. Deswegen entschieden wir uns für eine Approximierende Lösung der Koeffizienten durch Fully-Connected Layer. Um eine Vorhersage mehrerer diskreter Kategorien zu treffen, wird eine multinomial Regression benötigt.
Wir platzierten dazu Fully-Connected Layer in unterschiedlichen Tiefen der Netzwerke und ließen sie auf Spezies-,Genus-, und Familiennamen abbilden. Wir trainierten jedes Layer für ingesamt 10 Epochen ehe wir es auf dem PlantClef Testdatensatz von 2015 evaluierten. Das Preprocessing der Bilder ließen wir aus. Jedes Bild wurde auf eine Größe von 224×224 skaliert und standardisiert. Zur Standardisierung subtrahierten wir den Bildmittelwert und teilten durch die Standardabweichung. Die Trainingsparameter wählten wir wie in der vorherigen Trainingsphase (siehe 3.5). Als Fehlerfunktion benutzten wir erneut die Kreuzentropie. Im Gegensatz zur Evaluation im PlantClef Wettbewerb, bestand die Eingabe aus einem einzelnen Bild, was nur in seinen Farbwerten vorverarbeitet wurde. Die Vorhersage wurde nicht über diverse Bildausschnitte gemittelt. Wir protokollierten die *Top1 Genauigkeit*.

4.2.1 InceptionV1

Das InceptionV1 verschalteten wir nach allen Inception Modulen. Wir initialisierten alle restlichen Verbindungen mit den bereits trainierten Gewichten , welche während des Trainings konstant gehalten wurden. Lediglich die FC-Layer Verbindungen wurden optimiert.
Nachdem wir das Training auf Spezies-, Genus-, Familien- und Organ-Label abgeschlossen hatten, ließen wir das Netzwerk in unterschiedlichen Tiefen Vorhersagen für die Kategorien treffen. Abbildung 4.1 zeigt die Kategorie entsprechende *Top1 Genauigkeit* abgetragen über die Layer tiefe.

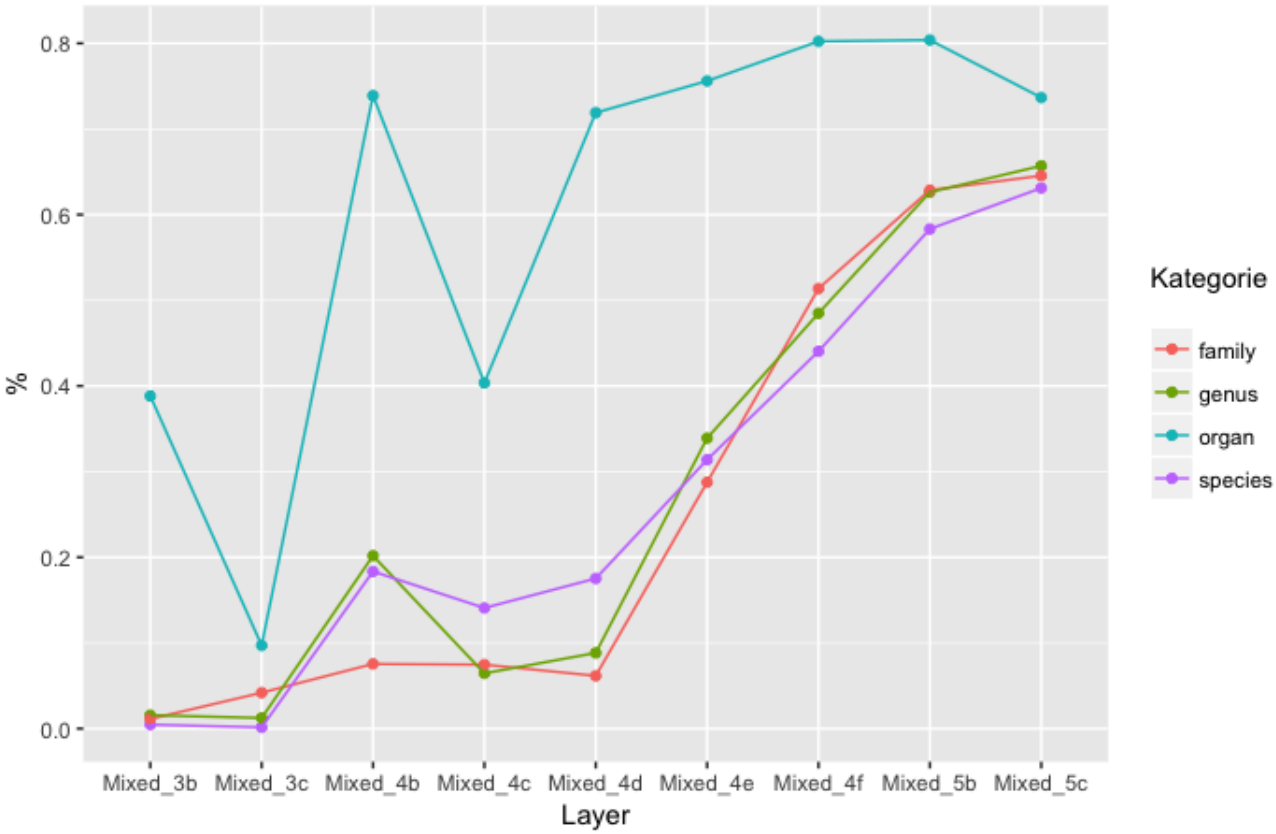


Abbildung 4.1: Entwicklung der Klassifikationsleistung (Top1 Genauigkeit) über die Tiefe des InceptionV1. Neben der bereits trainierten Vorhersage von Pflanzenspezies, wurde die Genauigkeit für Genus, Familie und Pflanzenorgan evaluiert.

Für Vorhersagen über das im Bild enthaltenen Pflanzenorgan erreicht das Netzwerk insgesamt die größte Genauigkeit von 80.3%. Die Genauigkeiten der Vorhersagen für Spezies, Genus und Familie befinden sich im Bereich von 63.1% (Spezies) bis 65.7% (Genus). Familien werden in 64.6% der Fälle richtig vom Netzwek erkannt. Die Ergebnisse auf Speziesebene sind schlechter als in der Replikation. Aufällig ist, dass die Leistung des Netzwerks für die Klassifikationen auf Organebene nach dem Inception Module 3c deutlich abnimmt, ehe sie im folgenden Layer auf fast 75% ansteigt. Ebenso fällt auf, dass nach dem Inception Module 4c die Leistung für alle Kategorien zusammenbricht. In folgenden Schichten steigt sie wieder an. Die Tiefe scheint insgesamt eine Rolle zu spielen. Bis auf die Kategorie Organ erreicht das Netz nach dem letzten Inception Module 5c seine höchsten Genauigkeitswerte. Für Organe zeigten die Ergebnisse höchst Werte nach dem vorletzen Inception Module 5b. Im letzten sinken sie hierfür um mehr als 5% ab.

Die schlechtere Leistung im Vergleich zur Replikation lßst sich durch das fehlende Mitteln über die Vorhersagen einzelner Bildausschnitte erklären.

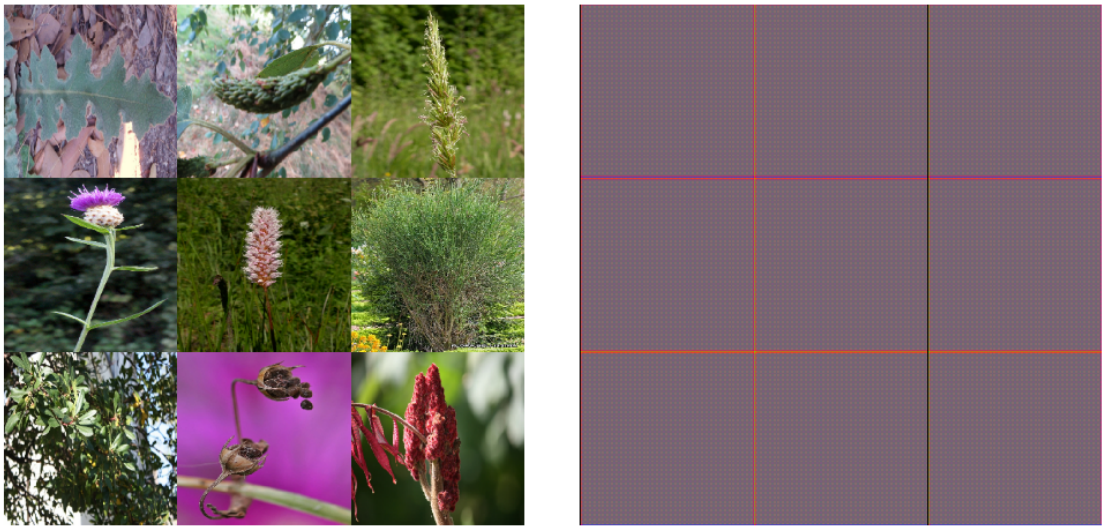
Score-Level-Averaging ist ein bekanntest Verfahren um die Vorhersagen eines neuronalen Netzes zu verbessern.
Die über die Layer deutlich bessere werdende Leistung legt nahe, dass die Tiefe des Netzwerkes eine entscheidende Rolle bei der Klassifizierung spielt. Unabhängig von der Kategorie der Klassen steigt die Genauigkeit ab dem Inception Module 4c an. Während sie für die Kategorien Spezies, Genus und Familie nach dem letzten Inception Modul 5c am grosten ist, sinkt sie für Organe ab. Wie bereits LeCun,Bengio und Hinton [Lecun et al., 2015] erkannten, lässt sich diese Beobachtung vermutlich auf das Lernen immer abstraktere Feature-Detektoren zurückzuführen. Tieferen Schichten besitzen in Bezug auf die Eingabe immer größere rezeptiven Fehler durch die sie auf komplexere Bildmerkmale reagieren. Diese nutzt das Netzwerk für eine spätere Klassifikation des Bildes. In unserem Fall wurde das Netzwerk auf die Klassifizierung von Pflanzenspezies trainiert, weswegen wir in spätere Layern pflanzenspezifische Filter vermuten. Bis zum Inception Module 5b scheinen diese Filter beschreibender Art zu sein und verschiedene Pflanzen Strukturen zu erkennen, ehe diese darauf organunabhängig kombiniert werden.
Vergleicht man die erreichten Genauigkeit des Netzwerks für die unterschiedlichen Kategorien, muss zunächst die Klassenanzahl berücksichtigt werden. Da die Chance richtig zu raten im Fall von 7 Klassen (Organ) im Gegensatz zu 1000 Klassen (Spezies) größer ist, subtrahieren wir die Zufallswahrscheinlichkeiten. Dabei ergaben sich Genauigkeiten von 63.0% (Spezies), 63.8% (Familie), 65.5% (Genus), 66.0% (Organ) für das letzte Inception Module 5c. Alle Kategorien werden mit vergleichbarer Genauigkeit klassifiziert, was ebenso für die Annahme spricht dass das Netz zunächst Bild beschreibende Merkmale wie Farbe, Ecken und Kanten in frühen bis hin zu Pflanzenstrukturen in späteren Schichten extrahiert. Die Filter im letzen Layer verknüpfen diese komplexere Pflanzenstrukturen. Da in den Eingabebilder verschiedene Pflanzenorgane enthalten können, müssen diese Filter organunabhängig. Nur so ist eine Zuordnung verschiedener Organe zu zusammengehörigen Pflanzenspezies möglich. Eine schlechtere Klassifikation auf Organebene ist die Folge.
Das Zusammenbrechen der Genauigkeit in mittleren Schichten (Inception-Module 4c) könnte unseren Überlegungen zu folge mit *co-adaptierten Filtern* zusammen hängen. In unsere Arbeit nutzen wir ein auf dem ImageNet vortrainiertes IncpetionV1 Modell. Die Schichten enthielten daher bereits Filter, die nun durch den neuen Datensatz angepasst werden sollten. Bereits Krizhevsky [Krizhevsky et al., 2012] und auch Zeiler und Fergus [Zeiler and Fergus, 2013] konnten zeigen, dass in frühen Schichten nach wenigen Epochen Filter konvergieren. *Low-level Featuredetektoren*, die sich unabhängig vom Datensatz entwickeln . Tiefere Schichten hingegen weisen deutlich komplexere Filter auf welche, vom Inhalt der Trainingsmenge abhängen. Jason Yosinski, Jeff Clune et al. [Yosinski et al., 2014] beschäftigten sich mit der Frage inwiefern auch komplexere Featuredetektoren generalisiert eingesetzt werden können. Dabei untersuchten sie inwiefern bereits gelernte Filter auf ein neues Netz übertragbar sind. Sie konnten zeigen, dass Transferlernen durch zwei Faktoren negativ beeinflusst wird. Zum einen durch die Spezialisierung von höheren

Filtern, zum anderen durch Optimierungsprobleme bezogen auf Co-Adaption von Featuredetektoren innerhalb der mittleren Schichten. Co-adaptierte Detektoren sind nur im Kontext mit anderen in der Lage Bildmerkmale zu erkennen. Durch das Training auf Pflanzenspezies spezialisierten sich in unserem Fall die Filter der tieferen Schichten. Unser Modell des InceptionV1 hatte keine zusätzlichen Ausgabeschicht mit Softmaxclassifier und somit kein verstärktes Fehlersignal. Aufgrund des Vanishing Gradients Problems wird das zurückpropagierte Fehlersignal von Schicht zu Schicht deutlich schwächer. Die schlechter Klassifikation nach dem InceptionModule 4c wäre durch nur teilweises Anpassen co-adaptierte Filter zu erklären. Von einander abhängige Filter wäre somit gespalten und funktionsunfähig.

Um diese Aussage zu überprüfen visualisierten wir einzelne Filter in den unterschiedlichen Layern des InceptionV1. Wir berechneten die neun stärksten Aktivitäten durch Aufsummieren über die gesamte Feature-Map. Die Aktivitäten projizierten wir mittels Deconvolution zurück auf das Eingabelayer. Für frühe schichten visualisierten wir ebenso das Eingabe Bild nach Anwendung der Layeroperation, was der direkten Filteraktivität entspricht. Aufgrund von Max-Pooling ist diese Form der Visualisierung für spätere Schichten nicht mehr interpretierbar.
Die Implementierung der Deconvolution bestand bereits in einem GitHub Beitrag von InFoCusp [1], welchen wir für unsere Zwecke angepasst verwendeten.

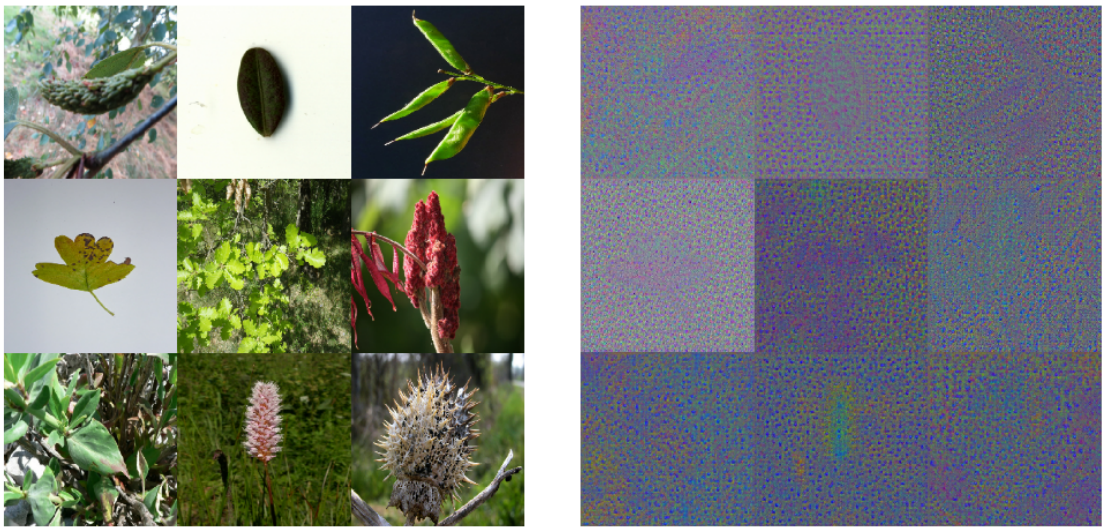
Bevor wir mit der Visualisierung begannen analysierten wir die Zusammensetzung des Datensatzes. 13367 (15%) Bilder zeigten Blätter, 12605 (13%) Blatt-Scans, 16236 (17%) die gesamte Pflanze, 5476 (5%) den Stamm, 8130 (9%) Äste, 28225 (30%) Bluten und 7720 (8%) Bilder von Früchten. Im ersten Convolutional Layer zeigen sich maximale Aktivitäten für farbige Gitterstrukturen. Der Filter in Abbildung 4.2 reagiert deutlich auf Rottöne. Im nächsten Convolution Layer erfolgt eine 1×1 Faltung mit anschließender 3×3 Faltung. Es sind komplexere Texturen in der aufgefalteten Feature-Map zu erkennen. Die Filter zeigen Aktivität für mehrfarbige Strukturen, was durch die 1×1 Convolution erklärbar ist. Durch sie werden einzelnen Filterdimensionen reduziert und auf eine Feature-Map abgebildet (Abbildung 4.3).

[1] https://github.com/InFoCusp/tf_ cnnvis (12 Juni 2017)



(a) Original Bilder (b) Feature-Map

Abbildung 4.2: Die Deconvolution der Feature Maps des ersten Convolutional Layers. Der Filter scheint deutliche auf rote Bildpixel zu reagieren.



(a) Original Bilder: Filter 22 (b) Feature-Map: Filter 22

Abbildung 4.3: Die Deconvolution der Feature Maps des zweiten Convolution Layer. Der Filter reagiert auf verschiedene Farbmuster.

Betrachtet man die Ausgaben der einzelnen Layer wird deutlich, dass sich *low-level Filter* gebildet haben. Abbildungen 4.4 zeigt am Beispiel eines Eingabebilds die direkte Filteraktivitt.

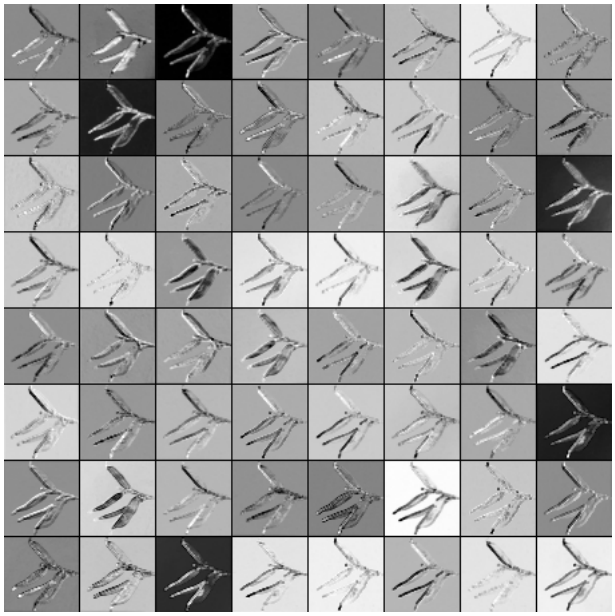
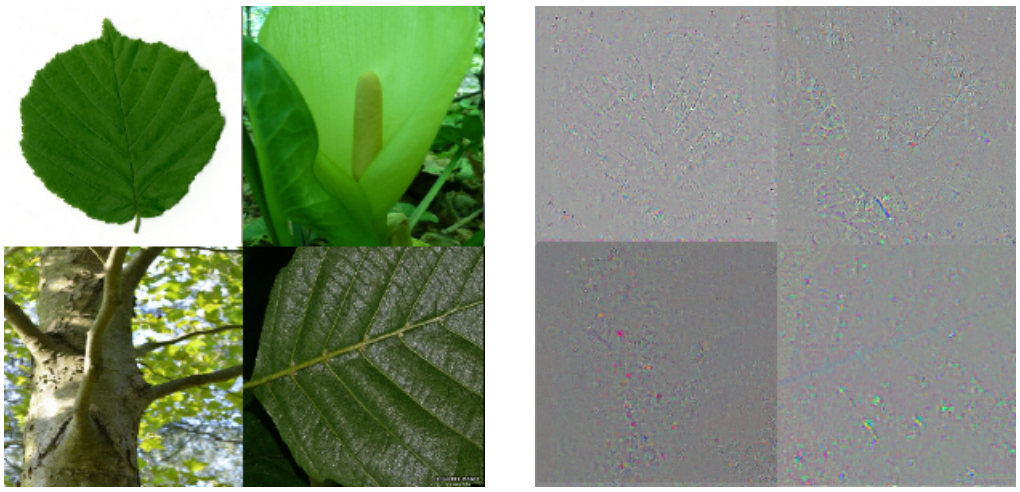


Abbildung 4.4: Eingabebild nach Durchführung der Faltung mit einem der 64 Filter des zweiten Convolutional Layer.

Darunter finden sich so genannte *Edge Detektoren*, die große Unterschiede zwischen Pixelwerten aufdecken. Angrenzende Pixel mit großem Intensitätsunterschied erscheinen weiß im bearbeiteten Bild, während Pixel mit ähnlicher Intensität dunkel abgebildet werden. Einige von ihnen scheinen orientierungsabhängig zu reagieren. Ebenso sind Filter erkennbar, die Bildelemente schärfen oder verwischen indem Intensitätsunterschiede verstärkt oder angeglichen werden.

Für die nachfolgenden Inception Module wählten Filter insgesamt 4 Filter aus unterschiedlichen Tiefen der Ausgabe. In diesen Modulen werden 1×1 und zwei 3×3 Faltungen parallel mit einer Max-Pooling Operationen durchführt. Für jede dieser Operationen entnahmen wir jeweils einen Filter und visualisierten die entsprechende Feature-Map. In den Mittleren Inception Modulen (3b,3c) deuten die Layerakivitäten weiterhin auf beschreibende Filter hin, welche auf Maserung und Muster reagieren (Abbildung 4.5). Durch 1×1 Faltung wird in einem kleinen Bereich über die gesamte Tiefe der vorangegangen Feature-Maps eine Aktivität bestimmt. Ohne dabei räumliche Informationen über angrenzende Bereich zu berücksichtigen, werden hier Bildmerkmale gesammelt. Bildbereiche in denen viele der vorherigen Filter Aktiv waren, zeigen Große Aktivitäten. Aktivitäten sind daher punktuell sehr stark. Die drastische Reduktion der Bildauflösung machen es schwer genauere Schlussfolgerungen aus den Visualisierten Aktivitäten zu treffen. Es fällt jedoch auf, dass das folgende Max-Pooling Layer unter den Top neun Aktivitäten zunehmend Blattscans, Äste oder Pflanzenstämme enthält. Es handelt sich um Großaufnahmen, in denen Maserungen und Konturen gut Erkennbar sind. Jedoch wird auch deutlich dass viele Aktivitäten durch Hintergrund Objekten ausgelöst werden. Ein Beispiel zeigt die Abbildng 4.6
Der Grund für den Einbruch der Klassifikationsleistung für Pflanzenorgane könnte damit zusammenhängen, dass einige Bilder des Datensatzes deutliches Hintergrundrauschen besitzen. Die Filter für komplexere Strukturen reagieren

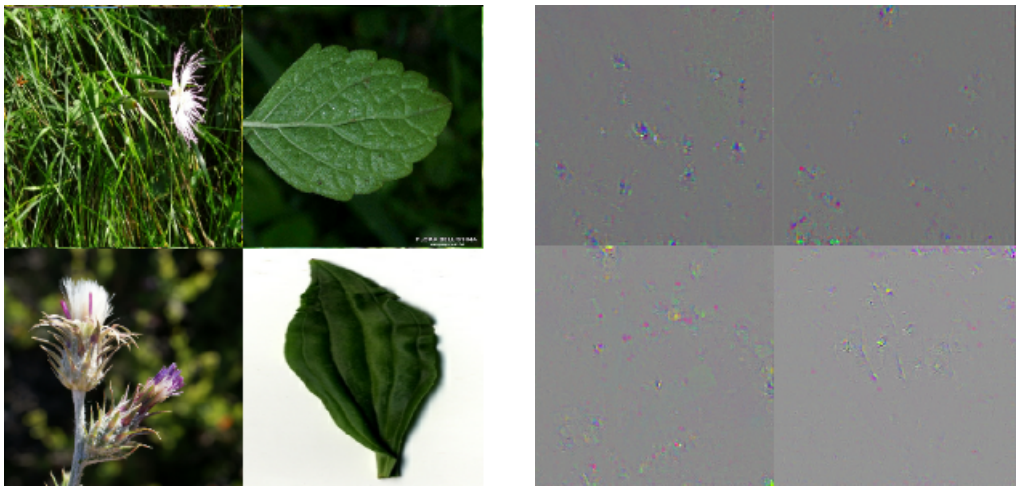
damit zunehmend auf das rauschen im Hintergrund. Bilder der Blüte und Frucht enthalten oftmals dieses Hintergrundrauschen in Form von umliegende Blätter und Verästlungen. Dadurch wird eine richtige Klassifizierung erschwert. Zwischen den Klassen Ast, Blatt, Blattscan, Stamm oder der ganze Pflanzen gibt es zudem auf der Ebene von komplexeren Mustern und Maserungen wenig Unterschiede.
Wir gehen davon aus, dass das Netzwerk auf Grundlage dieser Strukturen klassifiziert, daher nehmen wir an, dass 38% (Blüte und Frucht) der Bilder zunehmend falsch klassifiziert werden. Die restlichen 62% enthalten wenig Strukturelle Unterschiede und können daher nur auf Zufallsniveau klassifiziert werden. Für die restlichen 5 Klassen (Blatt, Blatt-Scan, Ast, Stamm, gesamte Pflanze) ergibt dass eine Genauigkeit von knapp 10%, welche auch in den Ergebnissen zu beobachten ist. Ebenso ist es möglich dass die schrumpfende räumliche Auflösung eine Rolle bei der Erkennung für Pflanzenorgane spielt. Während die Klassifikation für Pflanzenorgane deutlich einbricht steigt sie für Pflanzenspezies weiter an.



(a) Original Bilder (b) Feature-Map

Abbildung 4.5: Visualisierte Feature Maps des Incpetion Module 3c für Bilder mit größter Aktivität

Nach dem Inception Module 4b kommt es zu einem erneuten Anstieg der gesamten Klassifikationsleistung. Da die Deconvolution für dieses Layer Aktivitäten in sehr kleinen Bereichen des Bildes zeigt, vermuten wir, dass es sich um Filter handelt die auf feinere Muster und Strukturen reagieren, die es ermöglichen die Kategorien besser zu Unterscheiden. Die Auffaltungen einzelner Filter zeigten maximale Aktivität fü kleine Bereiche Nahe der Blüte, jedoch liß sich insgesamt wenig Information aus den Visualisierungen entnehmen, weswegen wir keine klare Aussage treffen können.
Ähnliches gilt für das nachfolgende Inception Module 4c. Hier kommt es zum einem Einbruch der Klassifikation für alle Kategorien. Wir betrachteten das Max-Pooling Layer des nachfolgenden InceptionModules 4d (Abbildung 4.7).



(a) Original Bilder (b) Feature-Map

Abbildung 4.6: Die Deconvolution der Feature Maps des Incpetion Module 4b . Gezeigt werden Bilder mit größter Aktivität in den Featuremaps verschiedener Filter.

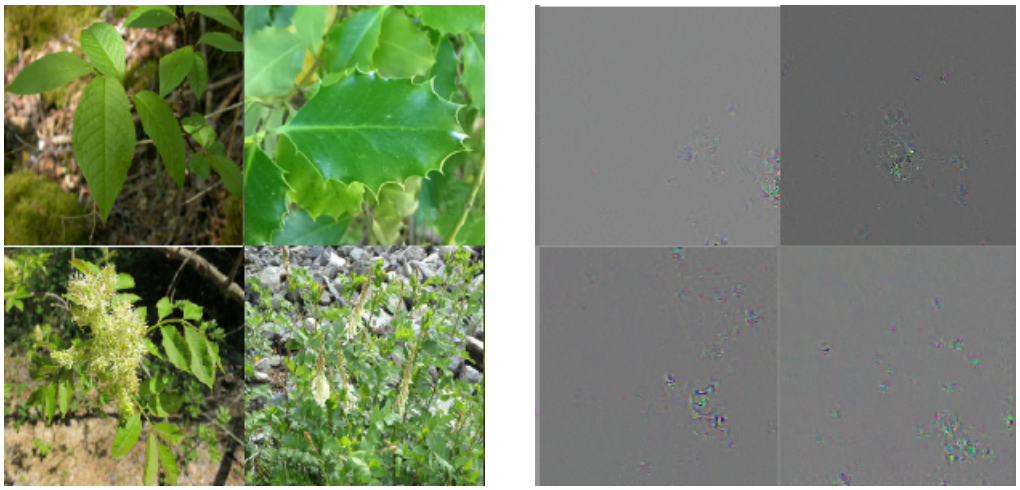
Genaue zusammenhänge der Aktivitätsmuster bleiben uns unersichtlich. Insgesamt geringere Aktivität über das Bild hinweg könnten für einen Rückgang der Filteraktivität im Inception Module 4c sprechen. Dies würde mit unserer Vermutung über teilweise funktionsunfähige Filter aufgrund von Coadaption übereinstimmen.

Über die nächsten Incpetion Module steigt die Genauigkeit des Netzwerkes für alle Kategorien weiter an. Wir visualisierten das verschiedene Layer im letzten Inception Module und versuchten aus den Bilder Erkenntnisse über die Maximale Aktivität der vorherigen Filter zu erlangen. Erneut sind es kleine Bereiche, die Maximale Aktivitäten hervorrufen. Bei genauere Betrachtung schließen wir, dass es feine Unterschiede in Form und Kontur einzelner Blättern sind, auf die der Filter reagiert 4.8.

Zum Schluss visualisierten wir die Aktivitäten in der Ausgabeschicht 4.9. Während im letzten Inception Module noch unterschiedliche Klassen die Filter stimulieren, werden nun die neun größten Aktivitäten von Bilder der selben Spezies ausgelöst. Darunter befinden sich sowohl Bilder von Blüte, Blatt, Frucht oder der gesamten Pflanze. Wie Abbildung 4.9 (d) zeigt, wird Aktivität durch Blattanordnung am Stiel der Pflanze, Blattform , Kelch und Kronblätter ausgelöst. Dies Aktivitäten zeigen verschiedene räumliche Ausdehnung.

(a) Original Bilder (Max-Pooling) (b) Feature-Map (Max-Pooling)

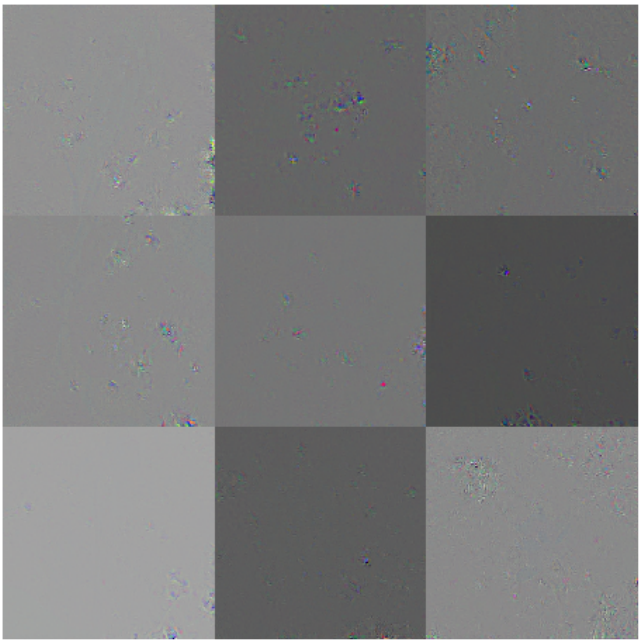
Abbildung 4.7: Die Deconvolution der Feature Maps des Incpetion Module 4b. Gezeigt werden Bilder mit größter Aktivität in den Featuremaps des Max-Pooling Zweigs. Die stärkst aktivierenden Bilder zeigen insgesamt wenig Filteraktivität



(a) Original Bilder (b) Feature-Map

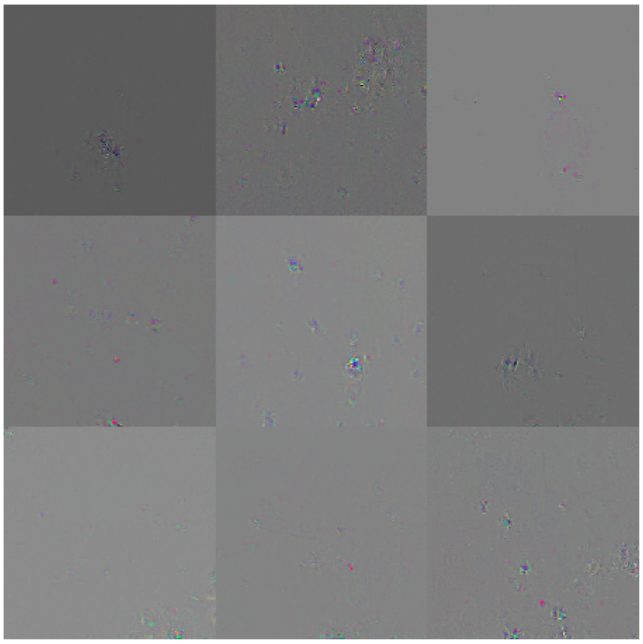
Abbildung 4.8: Die Deconvolution der Feature Maps des Incpetion Module 5c . Gezeigt werden Bilder mit größter Aktivität in den Featuremaps verschiedener Filter.

(a) Original Bilder: Catananche caerulea L.



(b) Feature-Map

(c) Original Bilder: Lactuca serriola L.



(d) Feature-Map

Abbildung 4.9: Die Deconvolution der Feature Maps der Ausgabe Schicht. Die dargestellten Bilder erzielten die größten werte für die Klassen Catananche caerulea L. und Klasse Lactuca serriola L. . Sie wurden alle richtig Klassifiziert.

Diese Ergebnisse zeigen auf, dass das Netz gelernt hat Pflanzenbilder anhand ihrer Bildmerkmale zu klassifizieren. Rückschlüsse aus Filteraktivitäten späterer Schichten waren schwer möglich, was wir auch die geringe Variation in unserem Datensatz zurückführen . Dennoch haben wir gesehen, wie in einzelnen Filter auf pflanzenspezifische Merkmale reagierten. Die Aktivitäten in frühen Inception Modulen sprechen für unsere Vermutung, dass Filter zunächst Muster detektieren. Diese werden über die Tiefe komplexer und sorgen im Bereich von Maserungen und verästelter Strukturen für stark aktivierte Feature-Maps. Später Schichten besitzen dagegen Aktivitätsmuster, die räumlich dichtere Strukturen zeigen. Zurückzuführen ist dieser spärliche Darstellung auf den Einsatz vieler 1x1 Filter innerhalb des Modelles. Spezielle Organdetektoren konnten unter den Filtern nicht erkannt werden, doch wurde anhand der Eingabebilder deutlich, dass vor allem Blattscans große Aktivitäten in den Filter auslösten.
Es kann in den Aktivitätsmustern eine Tendenz für Bestimmungsmerkmale erkannt werden (Abbildung 4.8). Die schnelle Reduktion der Bildauflösung durch häufiges Max-Pooling und Dimensions Reduktion durch 1×1 Filter (auch als Pooling der Feature-Maps zu sehen) machen es schwierig eindeutige Filter zu identifizieren.

4.2.2 Vgg16

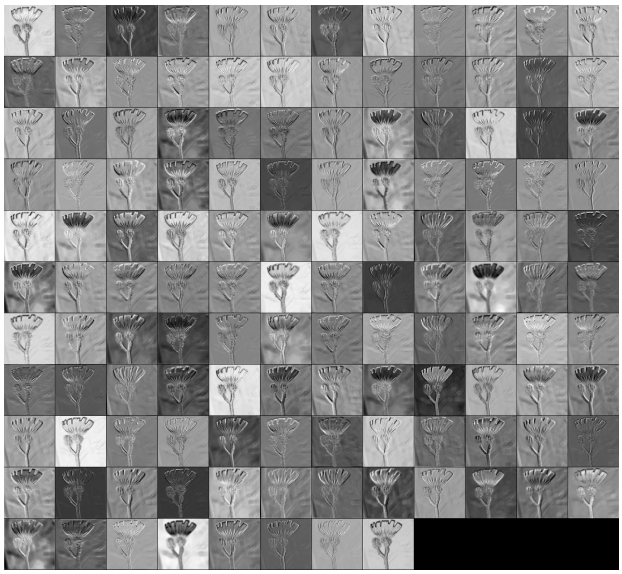
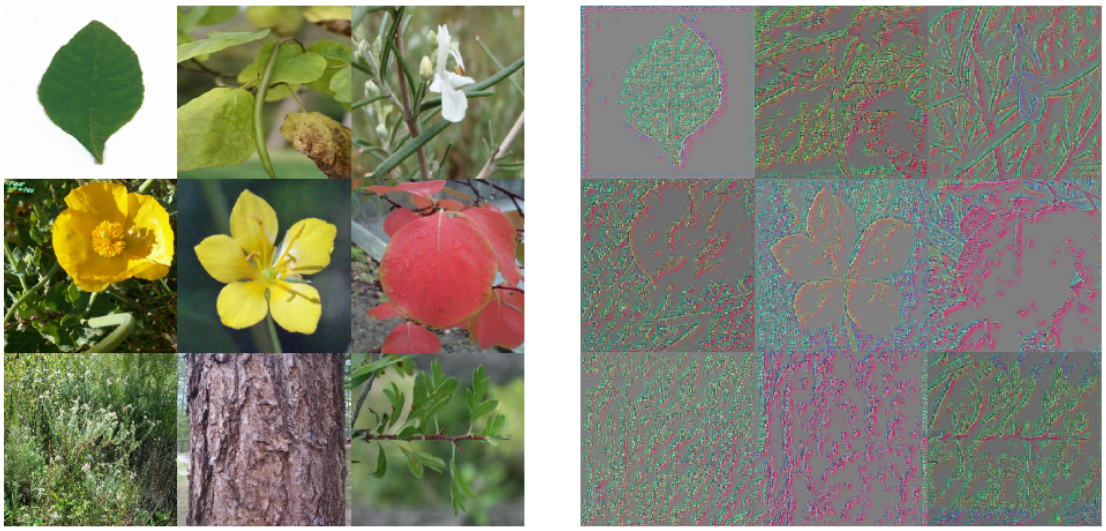


Abbildung 4.10: Visualisierte Filteraktivitäten des Convolutional Layer 2.1. Zu sehen ist das Eingabebild nach Durchführung der Faltung mit einem der 128 Filter.

Bei Fertigstellung unserer Arbeit befanden sich FC-Layer noch in der Trainingsphase, weswegen Aussagen und Vergleiche nur anhand der Visualisierten Aktivitäten getätigt werden konnten. Ebenso wie bereits beim InceptionV1 beobachtet, sind in den ersten Convolutional Layern und farb- und richtungssensitve Filter aufzufinden. 4.11. Abbildung**??** zeigt deutlich, wie kanten verstärkt oder abgeschwächt werden durch einzelne Filter.

Darauffolgenden Ebenen bilden mit zunehmender räumlicher Ausdehnung ihrer rezeptiven Felder komplexere Musterdetektoren. Insbesondere farbige Blüten und Blattscans sorgen für die größten Aktivitäten. Nach weiteren Convolutional Layer scheint das Netzwerk auf Blattformen zu reagieren. Während die aufgefaltete Feature-Map in Abbildung 4.12 (b) Aktivität für spitze Endungen zeigt, sind es Rundungen die Aktiviät in der Feature-Map in Abbildung 4.11 (d) auslösen. Die Blattform scheint wie bereits beim InceptionV1 ein Merkmal zu sein, welches das Netzwerk zu Klassifikation nutzt. Spätere schichten (Abbildung 4.13) zeigen, dass sich Filter gebildet haben die Organ unabhängig sind. Trotzdem ist wie bereits beim InceptionV1 eine klare Tendenz zu Blättern und Blüten zu erkennen. Die Visualisierung der Ausgabeschicht zeigt deutlich, dass das Netzwerk auf verschiedene Pflanzenbestandteile reagiert. Unterschiede zum InceptionV1 bestehen in der räumlichen Verteilung der Aktivitäten. Während in den Feature-Maps der Ausgabeschicht im Vgg16 größere zusammenhängende Bereiche erkennbar sind, besitzt das InceptionV1 kleinere, spärlichere Aktivitätsmuster. Man sieht zwei unterschiedliche Repräsentationen der Aktivitäten im Netzwerk, die beide gute Ergebnisse erzielen.

Nach Beenden unserer Analyse können wir sagen, dass für beide Netzwerke sich organunabhängige Filter finden lassen. Dennoch zeigen Aktivierungen in späteren Schichten eine klare Tendenz zu Blüten, Blattscans und einzelne Blätter (Abbildung **??**). Es scheinen sich in beiden Netzwerken vermehrt Filter

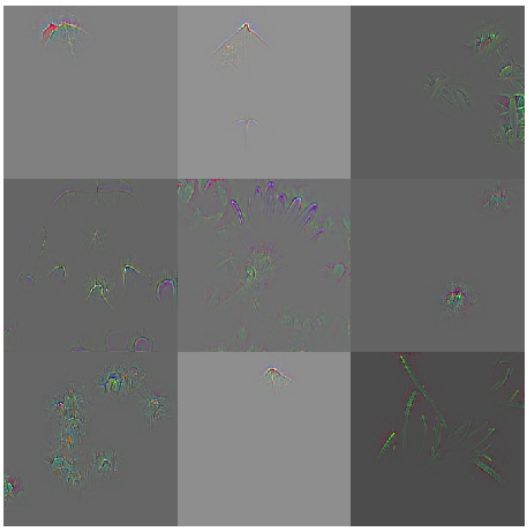


(a) Original Bilder: Max-Pooling Layer 2 (b) Feature-Map: Max-Pooling Layer 2

Abbildung 4.11: Die Deconvolution der Feature Maps das zweiten Max-Pooling Layer. Gezeigt werden Bilder mit größter Aktivität in den Featuremaps des Max-Pooling Zweigs

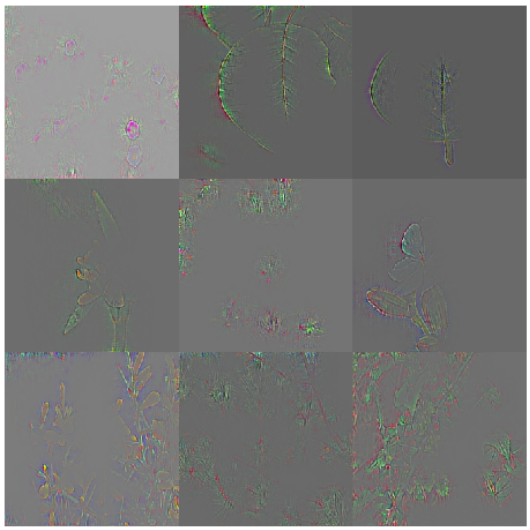
auf die Erkennung von Bildmerkmalen wie Maserungen, Konturverläufe oder Blattformen zu spezialisieren. Aus biologischer Sicht tragen Blätter und Blüten Informationen anhand derer im Pflanzenreich bereits viele Spezies Klassifiziert werden, weswegen eine Spezialisierung auf diese sinnvoll ist. Obwohl es keine klassische taxonomische Klassifizierung über die Tiefe des Netzwerkes stattfindet, zeigt das gelungene Transfer Lernen auf Genus- und Familienebene, dass sich biologisch sinnvolle Filter gebildet haben. Die Netzwerke lernen Bildmerkmale zu extrahieren, die eine Klassifikation auf allen Ordnungen ermöglicht. Abbildung 4.15 zeigt ein Beispiel in dem es zu Fehlklassifikationen kommt. Es enthält die neun höchsten Ausgaben des Vgg16 für die Spezies Vicia sepium L. (Genus: Vicia, Familie: Fabaceae). Darunter befinden sich falsch Klassifizierte Bilder des Astragalus glycyphyllos L. (Genus: Astragalus,Familie: Fabaceae) und Buglossoides purpurocaerulea (L.) I.M.Johnst. (Genus: Buglossoides, Familie: Boraginaceae) . Trotz Fehler scheint es die Familie richtig zu erkennen. Es lässt sich weiterhin sagen, dass Die Ergebnisse der Netze für unbekannte Bilder zeigen, dass Anhand von gebildeten Filter auch eine Generalisierung der Pflanzenklassifikation möglich ist.

(a) Original Bilder: Filter 229



(b) Feature-Map: Filter 229

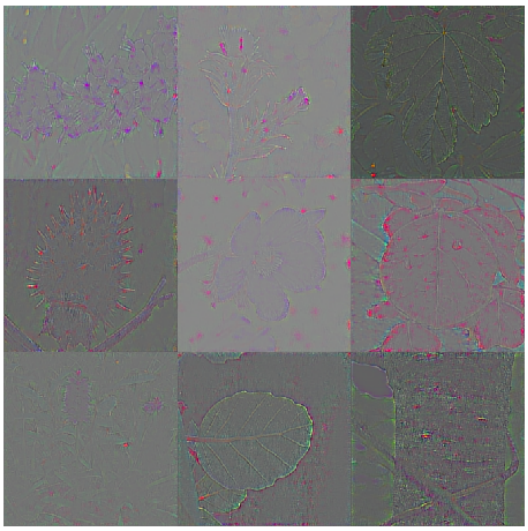
(c) Original Bilder: Filter 337



(d) Feature-Map: Filter 337

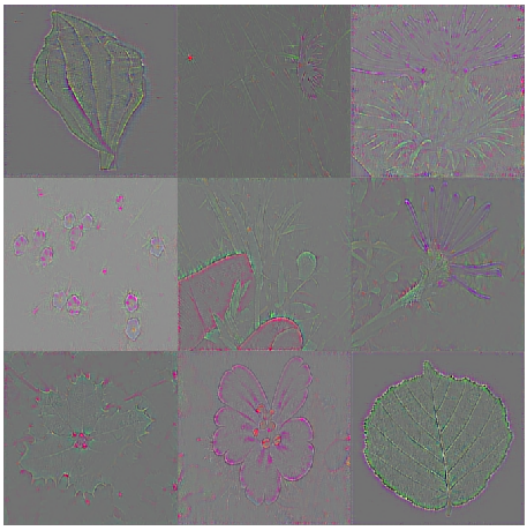
Abbildung 4.12: Die Deconvolution der Feature Maps des Vgg16 für das vierte Max-Pooling Layer. Gezeigt werden neun Bilder mit größter Aktivität in den Featuremaps der Filter 229 und 337.

(a) Original Bilder: Convolution Layer 5.1



(b) Feature-Map: Convolution Layer 5.1

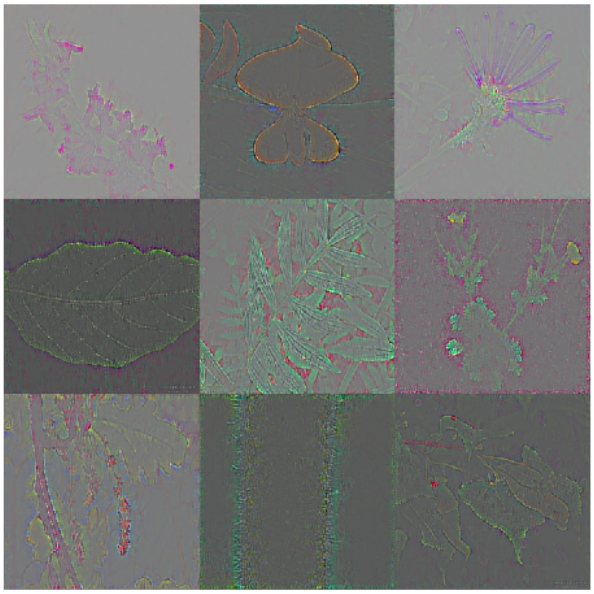
(c) Original Bilder: Convolution Layer 5.3



(d) Feature-Map: Convolution Layer 5.3

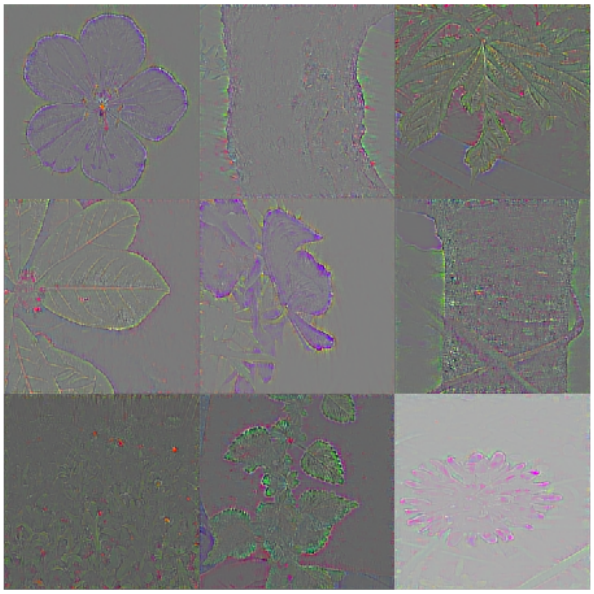
Abbildung 4.13: Die Deconvolution der Feature Maps des Vgg16 für die späten Convolution Layer 5.1 und 5.3. Es werden die neun Bilder mit größter Aktivität in den Feature Maps für zwei zufällig gewählte Filter gezeigt.

(a) Original Bilder: Geranium robertianum L.



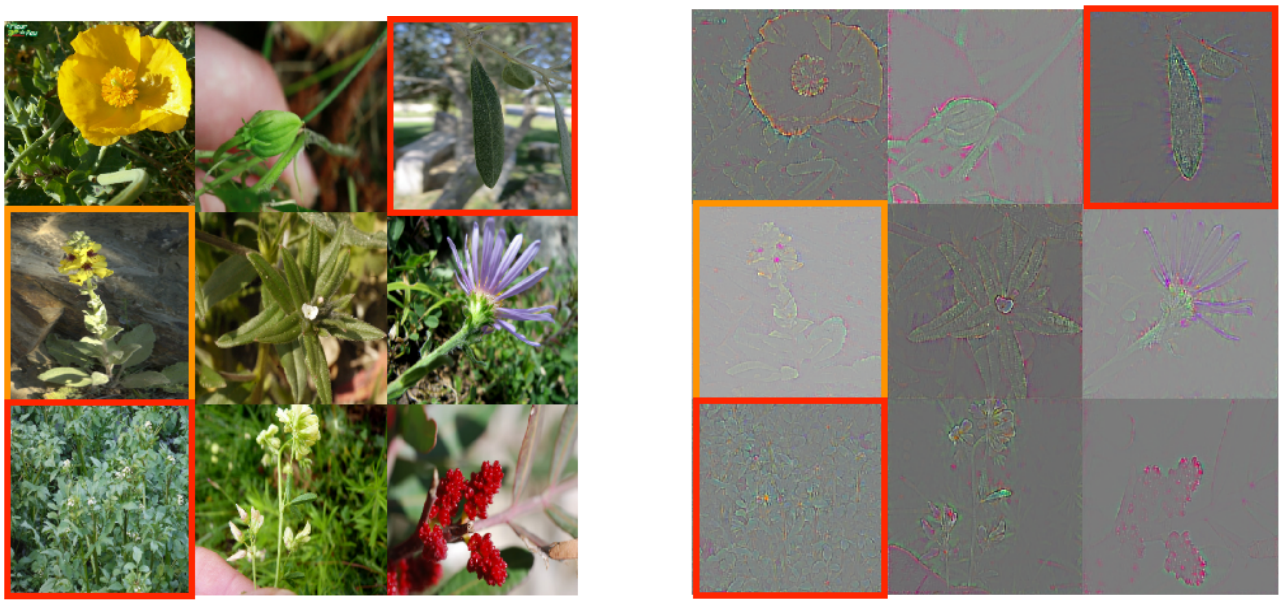
(b) Feature-Map

(c) Original Bilder: Glaucium flavum Crantz



(d) Feature-Map

Abbildung 4.14: Die Deconvolution der Feature Maps der Ausgabe Schicht. Die dargestellten Bilder erzielten die größten werte für die Klassen Geranium robertianum L. und Klasse Glaucium flavum Crantz . Sie wurden alle richtig Klassifiziert.



(a) Original Bilder: Vicia sepium L. (b) Feature-Map

Abbildung 4.15: Die Deconvolution der Feature Maps in der Ausgabe Schicht. Die dargestellten Bilder erzielten die größten Werte für die KlassenVicia sepium L. Umrandete Bilder wurden falsch Klassifiziert. Rot: Astragalus glycyphyllos L. ,Orange: Buglossoides purpurocaerulea (L.) I.M.Johnst.

Kapitel 5

Diskussion

Wie 3 zeigt, konnten wir die Ergebnisse von Ghazi et al. [Ghazi et al., 2016] und ihrem Sabanci Systems nur teilweise replizieren. Mit einem *MAP* von 67.5% (siehe **??**) blieb unser Ensemble unter den im Wettbewerb erreichten 73.8%. Auch die im Closed World Szenario, in dem nur Bilder von gelernten Spezies enthalten waren, oder bei der Klassifikation inverser Spezies, erzielte unser System schlechtere Ergebnisse. Wir erreichten 61% auf den Invasiven Pflanzenbildern und 72.7% im closed World Szenario. Wir verwarfen dabei 1236 Bilder als nicht-pflanzlich und 1434 aufgrund niedriger Ausgabewerte.
Als wir den Schwellenwert vernachlässigten, konnten wir unsere Resultate verbessern. Mit 70.7% im closed World Szenario, 0.641% auf den Invasiven Pflanzenbilder und 70.7% erreichten wir vergleichbare Ergebnisse. Durch das Binäre InceptionV1 verwarfen wir $1236 Bilder$ als nicht pflanzlich. Diese Ergebnisse sprechen dafür dass unsere Netzwerke unterschiedliche Vorhersagen für einzelne Bilder trafen.
Durch den zusätzliche Schwellenwert wurden zusätzlich Bilder verworfen deren größten Ausgabewerte unterhalb des Schwellenwertes lagen. Dies hatte in unserem Fall zu folge, dass zusätzlich schlecht erkannte Pflanzenbider aussortiert wurden. Bei Vernachlässigung des Schwellenwertes, flossen alle pflanzlichen Bilder in die Metrik mit ein. Auch wenn sie schlechte Ergebnisse erzielten, hatten so Bilder von bekannten Objekten größeren positiven Einfluss auf die Metrik als Bilder von Unbekannten Klassen negativen. Diese Erkenntnis lässt zweifel an der Metrik aufkommen.
Wir vermuten, dass ein Faktor fr die aufgetretenden Unterschied ein Fehler bei der preprocessing Methodik war. Anstatt das Bild um $R°$ in beide Richtungen zu drehen, drehten wir es nur in eine. Wir entnahmen aus der gedrehten Variante einen zentralen Bildausschnitt. Ausschnitt und gedrehtes Bild fügten wir unserer Liste hinzu, aus der wir anschließend zufällig Bilder zogen. Wir erkannten und behoben diesen Fehler erst nach dem die Trainingsphase abgeschlossen war. In nachfolgenden Evaluation nutzten wir die fehlerfreie Methode und mittelten Netzwerkvorhersagen über entsprechende Bildvarianten. Dadurch traten Unterschiede zwischen gelernten Bildausschnitten und den bei Evaluation generierten auf, welche mögliche Auswirkung auf die Genauigkeit unserer Netzwerke hatten. Nichts desto trotz verlief das Training problemlos und wir konnten gute Ergebnisse auf Validierung und Testdaten

erzielen. Damit konnten wir zeigen dass unsere Netze eine Generalisierung lernten. Sowohl das InceptionV1 als auch das Vgg16 scheinen somit beide für die Klassifikation von Pflanzen geeignet zu sein.

Das Trainieren zusätzlicher FC-Layer auf unterschiedliche Kategorien zeigte uns, dass die Tiefe eines Netzwerks eine entscheidende Rolle spielt. Unabhängig von Kategorie werden in tieferen Schichten bessere Ergebnisse erzielt. Diese Erkenntnis spiegelt den Ansatz wieder, den Simonyan und Zisserman [Simonyan and Zisserman, 2014], und Szegedy et a.[Szegedy et al., 2014] bei der Entwicklung ihrer Netzwerkarchitekturen verfolgten. Ein bekanntest Problem tiefer Netze besteht durch das Vanishing Gradient Problem, welches auch wir beobachten konnten. Unsere Einbrüche der Klassifikationsleistungen des InceptionV1 führen wir auf selbiges Problem zurück. Wir verwendeten in unserer Implementation keine zusätzlichen Ausgabeschichten um Fehlersignale zu verstärken. Dadurch fanden Gewichtupdates durch Pflanzenbilder nur in tieferen Schichten. Frühere Schichten hingegen blieben nahezu unverändert, was womöglich negative Auswirkung auf co-adaptatierte Filter hatte.
Der Verlauf der Klassifikationsleistung legt nahe, dass frühe Schichten deskriptive Filter besitzen und erst in späteren Schichten diskriminierende Filter entstehen. Deskriptoren helfen bei der Unterscheidung von Organen. Für spezielleren Aufgaben wie Klassifikation von Spezies, Genus oder Familien sind pflanzenspezifischere Filter notwendig, welche sich erst in späteren Schichten ausbilden.
Der Vergleich der Leistung für Kategorien deutet darauf hin, dass diese Filter keine taxonomische Ordnung besitzen. Yosinski et al.[Yosinski et al., 2014] erkannten bereits 2014, dass in frühen Schichten Filter entstehen, die generell Einsetzbar sind . Spätere Schichten besitzen hingegen immer spezieller, auf die Aufgabe abgestimmt Filter. Diese können nur sehr schwer für andere Aufgaben eingesetzt werden.
In unserem Versuch trainierten wird die Netze Spezies zu klassifizieren. Die Transferaufgabe,die das Netzwerk leisten sollte, war die Klassifizierung von Familien und Geni, welche es ohne weiteres Training tieferer Schichten schaffte. Wir schließen daraus dass das Netzwerk Vorhersagen aufgrund von sinnvollen, pflanzen spezifischen Filter trifft, die es über die Tiefe der Layer lernt. Das Auftreten verschiedener extrahierter Merkmale wird in der letzten Schicht genutzt, um daraus eine Ausgabe zu berechnen.

Auch aus der Visualisierungen ist ersichtlich, dass frühe Schichten low-level Featuredetektoren enthalten. Wie bereits Krizhevsky oder auch Zeiler und Fergus [Krizhevsky et al., 2012, Zeiler and Fergus, 2013] zeigen konnten handelt es sich dabei um Gabor ähnliche Filter. Sie extrahieren generelle Bildmerkmale und sind über verschiedene Datensätze und Aufgaben einsetzbar. Da unsere Modelle auf vortrainierten Versionen aufbauen, gehen wir davon aus, dass diese bereits bestanden und durch die Daten nicht weiter angepasst wurden. Die Erregungsmuster dieser Schichten zeigen typische Aktivitäten für Farbe und natürliche Muster. Die Visualisierungen der Eingabe Bilder nach Anwendung der Filter waren für frühe Schichten aussagekräftig. Danach

konnten wir sie nicht mehr interpretierbar, weswegen nur noch Aussagen auf der Grundlage der Feature-Map Aktivität getroffen wurden. Diese ließen sich aufgrund geringer räumlicher Auflösung im InveptionV1 und der ähnlichkeit der Daten schwer deuten. Dennoch erkannten wir vereinzelt Zusammenhänge in den Aktivitätsmustern und zeigten einige Beispiele auf, in denen auf mögliche Bestimmungsmerkmale reagiert wurde. Diese Merkmale lassen sich sowohl für die Klassifikation von Pflanzenspezies, Genus oder Familie nutzen. Wir konnten auch hier sehen, dass die Klassifizierung der Neuronalen Netze keine taxonomische ist. Das Netzwerk scheint über die Tiefe Filter gelernt zu haben, welche generelle Pflanzenstrukturen segmentieren. Diese Pflanzenbestandteile werden durch immer komplexere Feature-Detektoren dem Bild entnommen.Wir konnten anhand der aufgefalteten Feature-Maps Aktivitäten in Bereichen erkennen, die Bestimmungsmerkmale enthielten. Spezifische Filter für Pflanzenorgane waren nicht zu erkennen. Grund dafür sehen wir in der Ungleichverteilung im Datensatz und dem starken Hintergrundrauschen. Einzelne Filter des Vgg16 4.13 zeigten jedoch unter ihren top-9 Aktivitäten Tendenzen für einzelene Organe. Unter anderem für Blattscans. Eine bessere Verteilung innerhalb der Bilddaten könnte spezialisierte Filter entstehen lassen, welche zu einer Verbesserung der Leistung beitragen könnten.

Ähnlicher Ansatz wird in Applikationen wie Leafsnap genutzt [Kumar et al., 2012]. Dieses Erkennungstools verwenden Fotos von Blättern mit weißem Hintergrund und erreichen damit gute Ergebnisse. Auch Lee, Chang et al. [Lee et al., 2015] zeigten in ihren Versuchen von 2015, dass die Information über das zu sehende Pflanzenorgan die Klassifikationsleistung ansteigen lassen. Ihr Netzwerk kombiniert generelle Bildinformationen mit extrahierte Informationen über Pflanzenorgane. Dazu werden auf allgemeine Klassifikation und speziell auf Organ Klassifikation trainierte Schichte in einem gemeinsamen Netz verwendet. Die Bildinformation wird parallel verarbeitet und späteren zusammengeführt. Mit diesem Anstatz erreicht das Team um Lee gute Ergebnisse. Interessant wäre daher zu sehen, ob ein weiteres paralleles Netz zur Organklassifizierung Vorteile auf Speziesebene bringen würde.

Weiterhin sehen wir Potential für Verbesserungen in der Nutzung von Kontextinformation. Über die im Datensatz enthaltene XMl Datei können Informationen über den Vorkommensort der Pflanze entnommen werden, welche die zu Klassifizierenden Arten einschränkten könnten. Ebenso zeigt der Vergleich zu einer parallel durchgeführten Bachelorarbeit [Beller, 2017] zeigt, können neben anderer Architekturen auch Preprocessing Maßnahmen für eine Verbesserung sorgen. Unsere Idee war es diesbezügliche das Spiegeln der Bilder auszulassen, um Informationen über Blattanordnungen zu erhalten. Da jedoch ein Fehler in der Methode gemacht wurde, können wir keine Aussage über einen möglichen Nutzen treffen.

Um besseres Verständnis über die internen Vorgänge der Netze zu erlangen nutzten wir den Ansatz eines Deconvolutional Networks. Die visualisierten

Feature-Maps für spätere Schichten ließen jedoch nur bedingt Aussagen über die extrahierten Bildmerkmale zu. Ein weiterers Ansatz Bildbereiche einzugrenzen wäre daher eine Heatmap zu erstellen [Zeiler and Fergus, 2013]. Bereiche größter Aktivität können und das Verdecken einzelner Bildstellen könnte zusätzliche Auskunft über komplexe rezeptive Felder geben und womöglich besser Interpretierbar sein.
Interessant wäre ebenso die Erstellung von Urbildern als Analyseansatz . Bei Eingabe eines Bildes berechnet das Netzwerks Ausgaben für jede Klasse. Die Werte werden dabei durch eine Softmax-Funktion $\sigma(x_j) = \frac{e^x_j}{\Sigma_{k=1}^{K} e^{x_k}}$ bestimmt, die sie auf eine Zahl zwischen 0 und 1 abbildet. Optimal wäre eine Berechnung in der 1 für die richtige Klasse j und 0 für alle anderen Klassen $k \neq j$ ausgegeben werden. Eingaben die eine 1 erzeugen würden wären damit Repräsentanten der Klasse. In der Praxis ist dies fast nie der Fall jedoch können solche Eingaben künstlich erzeugt werden. Der Ansatz der innerhalb der Google Research Abteilung [Mordvintsev et al., 2015] entwickelt wurde, besteht darin ein Eingabebild zu produzieren, welches einen Ausgabewert von 1 für die entsprechende Klasse besitzt. Dazu wir ein Bild,was zunächst aus Rauschen besteht, mittels Gradientaufstieg wird jeder Pixelwert so verändert, dass es die Klassenaktivität maximiert. Die veränderte Eingabe stellt das Urbild der Klasse dar. Anhand diese Urbilds können klassenspezifische Merkmale erkannt werden, die das Netzwerk gelernt hat. Um noch klarere Bilder zu erlangen können Regulierungs Methoden wie Laplace-Pyramiden-Gradienten-Normalisierung angewendet werden. Das Einbringen von Apriori wissen, wie die Korrelation benachbarter Pixel, verbessern das Verfahren zusätzlich . [Yosinski et al., 2015]

Zum Abschluss dieser Bachelorarbeit möchten wir sagen, dass im Laufe unsere Arbeit viele Ansätze auftauchten, die aus zeittechnischen Gründen nicht weiter verfolgt werden konnten. Wir erlebten die zeitlichen Ansprüche, die ein gut Trainiertes Netz stellt und konnten sehen wie leistungsabhängig es von Daten ist.
Wie sind der Meinung, dass Convolutional Networks ein enormes Potential bei der Objektklassifizierung besitzen und viel Raum für weiter Forschung bieten. Verbesserungen werden jedoch oftmals nur durch erhöhte Rechenleistungen und größere Datenmengen erzielt. Das in unserer Arbeit replizierte System ist dafür ein Beispiel. Durch zusammenschalten mehrerer Architekturen und Datensatz Vergrößerung konnten gute Ergebnisse erzielt werden.
Jedoch zeigen beispiele wie die Versuche von Ngyuen, Yosinski und Clune [Nguyen et al., 2014] die Schwächen von Neuronalen Netzen. Durch leichte für das menschliche Auge nicht wahrnehmbare Veränderung der Bilder, können die Ausgaben eines Netzwerkes deutlich verändert werden. Deswegen sind auch wir der Überzeugung von Matthew Fergus und Rob Fergus, dass ohne eine Verständnis der Vorgänge innerhalb eines neuronalen Netzes und ohne biologisch plausible Ansätze, Verbesserungen ein reines Trial and Error Spiel sind.

Literaturverzeichnis

[Arora et al., 2013] Arora, S., Bhaskara, A., Ge, R., and Ma, T. (2013). Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343.

[Beller, 2017] Beller, M. (2017). Tiefe residuale netzwerke zur pflanzenklassifikation. *Bachelor Thesis Universitt Tbingen*.

[Bendale and Boult, 2015] Bendale, A. and Boult, T. E. (2015). Towards open set deep networks. *CoRR*, abs/1511.06233.

[Ciresan et al., 2011] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, pages 1237–1242.

[Clough and Sanderson, 2003] Clough, P. and Sanderson, M. (2003). The clef 2003 cross language image retrieval track. *Cross Language Evaluation Forum (CLEF)*.

[Gabor, 1946] Gabor, D. (1946). Theory of communication. *J. Inst. Elec. Eng. (London)*.

[Ghazi et al., 2016] Ghazi, M. M., Yankolu, B., and Aptoula, E. (2016). Open-set plant identification using an ensemble of deep convolutional neural networks. *CLEF 2016 Conference*.

[Goau et al., 2016] Goau, H., Bonnet, P., , and Joly, A. (2016). Plant identification in an open-world (lifeclef 2016). *CLEF 2016 Conference*.

[Hang et al., 2016] Hang, S. T., Tatsuma, A., and Aono, M. (2016). Bluefield (kde tut) at lifeclef 2016 plant identification task. In *CLEF*.

[Hebb, 1949] Hebb, D. (1949). *The organization of behavior. A neuropsychological theory. , ISBN 0-8058-4300-0 (Nachdruck der Ausgabe New York 1949)*. New York: John Wiley & Sons, Inc.

[Hubel and Wiesel, 1959] Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Physiological Society*.

[Hubel and Wiesel, 1962] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Physiological Society*.

[Johnson et al., 1991] Johnson, M. H., Dziurawiec, S., Ellis, H., and Morton, J. (1991). Newborns preferential tracking of face-like stimuli and its subsequent decline. *Cognition. 40*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

[Kumar et al., 2012] Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., and Soares, J. V. B. (2012). *Leafsnap: A Computer Vision System for Automatic Plant Species Identification*, pages 502–516. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Lecun et al., 2015] Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

[Lee et al., 2015] Lee, S. H., Chan, C. S., and Remagnino, P. W. P. (2015). Deep-plant: Plant identification with convolutional neural networks. *IEEE*.

[Marĉelja, 1980] Marĉelja, S. (1980). Mathematical description of the responses of simple cortical cells∗. *J. Opt. Soc. Am.*, 70(11):1297–1300.

[Mordvintsev et al., 2015] Mordvintsev, A., Olah, C., and Tyka, M. (2015). Inceptionism: Going deeper into neural networks. `https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[Nguyen et al., 2014] Nguyen, A. M., Yosinski, J., and Clune, J. (2014). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Nature*.

[Russakovsky et al., 2012] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2012). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*.

[Scheirer et al., 2014] Scheirer, W. J., Jain, L. P., and Boult, T. E. (2014). Probability models for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36.

[Sermanet et al., 2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

[Serre et al., 2007] Serre, T., Wolf, L., Bileschi, S. M., Riesenhuber, M., and Poggio, T. A. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):411–426.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Researc*.

[Śulc et al., 2016] Śulc, M., Mishkin, D., and Matas, J. (2016). Very deep residual networks with maxout for plant identification in the wild. *CLEF 2016 Conference*.

[Szegedy et al., 2014] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

[Werbos, 1974] Werbos, P. J. (1974). *Beyond regression : New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University.

[Werbos, 1988] Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356.

[Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3320–3328, Cambridge, MA, USA. MIT Press.

[Yosinski et al., 2015] Yosinski, J., Clune, J., Nguyen, A. M., Fuchs, T. J., and Lipson, H. (2015). Understanding neural networks through deep visualization. *CoRR*, abs/1506.06579.

[Zeiler et al., 2011] Zeiler, M., Taylor, G., and Fergus, R. (2011). *Adaptive deconvolutional networks for mid and high level feature learning*, pages 2018–2025. IEEE.

[Zeiler and Fergus, 2013] Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift