Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät

Bachelorarbeit Kognitionswissenschaft

Tiefe Residuale Netzwerke zur Pflanzenklassifikation

Maximilian Beller

Matrikelnummer 3951415

19.08.2017

Gutachter

Prof. Dr. Hanspeter A. Mallot
Institut Neurobiologie - Kognitive Neurowissenschaften
Eberhard Karls Universität Tübingen

Betreuer

Gerrit Ecke
Institut Neurobiologie - Kognitive Neurowissenschaften
Eberhard Karls Universität Tübingen

Beller, Maximilian:
Tiefe Residuale Netzwerke zur Pflanzenklassifikation
Bachelorarbeit Kognitionswissenschaft
Eberhard Karls Universität Tübingen
Bearbeitungszeitraum: 19.04.2017 – 19.08.2017

Erklärung

Hiermit erkläre ich, dass ich diese schriftliche Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe.

Ort, Datum Unterschrift

1. Abstract

Im Rahmen dieser Bachelorarbeit wurde das tiefe Residuale Neuronale Netzwerk ResNet-50 mit MaxOut Schicht zur Klassifikation von Pflanzen trainiert. Anschließend wurde untersucht, ob das Netzwerk die Taxonomie von Pflanzen gelernt hat und ob sich sinnvolle rezeptive Felder gebildet haben. Hierzu wurde ein Softmax-Klassifikator auf den Ausgaben der vier Blöcke des Residualen Netzwerkes trainiert. Des Weiteren wurden die rezeptiven Felder auf diesen Ebenen mit Hilfe eines *Deconvolutional network* visualisiert.
Das Netzwerk erreicht Klassifikationsleistungen von bis zu 64.48% auf unbekannten Daten. Die Netzwerkanalysen ergaben, dass bereits frühe rezeptive Felder Blüten von anderen Pflanzenorganen unterscheiden können. Weitere Pflanzenorgane erkannte das Netzwerk jedoch auch in späteren Schichten nicht. Die Erkennungsrate für die Familie, Gattung und Spezies der Pflanze steigt erst durch den vierten Block des Netzwerkes an. Diese Ergebnisse sprechen dafür, dass sich dort diskriminierende Filter gebildet haben, welche sich von den Filtern früher Schichten unterscheiden. Die frühen Filter kodieren meist nur deskriptive Informationen.

Inhaltsverzeichnis

2. Einleitung

Nach Erkenntnissen von Riesenhuber und Poggio (1999) sowie Serre et al. (2005) sind die ersten 100-200ms der Verarbeitung im ventralen Pfad des visuellen Kortex von Primaten feed-forward (Serre, Wolf, Bileschi, Riesenhuber & Poggio, 2007). Das bedeutet, dass es keine Rückkopplungen in einem hierarchisch aufgebauten System gibt. Auch Thorpe, Fize und Marlot (1996) berichten von Objekterkennungszeiten unter 150ms. Serre et al. (2007) nutzen diese Erkenntnis um ein biologisch inspiriertes Model zur Objekterkennung zu entwerfen. Als die wichtigsten Eigenschaften dieses Models bezeichnen sie die *hierarchische* Aufbauweise, welche zuerst Unabhängigkeit von der Position und Größe des Bildinhaltes und später auch vom Blickwinkel schafft. Als zweite wichtige Eigenschaft werden die *rezeptiven Felder* genannt, deren Komplexität immer weiter steigt. Das rezeptive Feld eines Neurons bestimmt gewissermaßen auf welche Reize dieses Neuron reagiert und auf welche nicht. So kann ein rezeptives Feld sowohl ein Kantendetektor, als auch ein Gesichtsdetektor sein. Des Weiteren werden Rückkopplungen aufgrund der beschränkten Zeit ausgeschlossen, es handelt sich um ein *vorwärts gerichtetes System*. Die Autoren nehmen weiterhin an, dass *Lernvorgänge* und *plastische Veränderungen* des Systems in allen Verarbeitungsebenen vorkommen.

Neuronale Netze, welche heutzutage die besten maschinellen Ergebnisse bei Aufgaben des Bildverstehens erzielen, sind ebenfalls nach diesen Kriterien aufgebaut. Ein Neuronales Netz besteht aus sogenannten Neuronen. Ein Neuron erhält eine Eingabe x, berechnet auf dieser Eingabe seine Aktivität und gibt eine Ausgabe $f(x)$ weiter. Diese Berechnungen sind *nicht-linear*, das bedeutet die Gleichung $\forall x : f(x) = m \cdot x$ ist nicht erfüllt. Neurone können auf verschiedene Art und Weise miteinander verbunden werden. Die hier betrachtete Architektur ist die eines vorwärts gerichteten Netzes. Dabei sind Neuronen einer Schicht zugeordnet und mit allen Neuronen der nächsten und vorherigen Schicht über gewichtete und vorwärts gerichtete Verbindungen gekoppelt. Die Gewichte dieser Verbindungen können über sogenannte Lernregeln verändert werden (Mallot, 2013, S. 82-83). Zwar können durch diese Lernregeln keine neuen Verbindungen entstehen oder alte gelöscht werden, jedoch können die Gewichte auf Null reduziert, oder von Null auf einen anderen Wert verändert werden. Dadurch werden die biologischen Vorgänge simuliert.
Der Input eines Neurons ist die gewichtete Summe aller Ausgaben der Neurone der vorherigen Schicht, gepaart mit einem Schwellenwert θ. Für ein Neuron j mit der nicht-linearen Aktivitätsfunktion f(x), berechnet sich dessen Aktivität o_j somit anhand der Formel

$$o_j = f(\Sigma_i w_{ij} o_i - \theta_j).$$

Dabei ist w_{ij} das Gewicht, welches Neuron i und Neuron j verbindet. Durch die Berechnungen vieler hintereinander geschalteter Schichten von Neuronen werden die Dimensionen der Eingabe immer weiter verändert, bis es möglich ist, die Daten linear zu separieren. So können beispielsweise Bilder, welche diskreten Gruppen zugeordnet sind, unterschieden werden. Dabei wird der zu Beginn hoch dimensionale Raum, der sich aus der Höhe und Breite, sowie der Anzahl der Farbkanäle (Rot, Grün, Blau) zusammensetzt, soweit verändert bis eine Zuordnung stattfinden kann. Diese Aufgabe wird als Bildklassifikation (engl. *image classification*) bezeichnet und ist eine der Teilgebiete des maschinellen Sehens (engl. *computer vision*).
Während des Trainings, also der Phase in der die Gewichte des Systems verändert werden, wird dem Klassifikationssystem sowohl das Bild, als auch die zugehörige Klasse präsentiert. Das Neuronale Netz gibt nun an zu welcher der diskreten Klassen es das Bild zuordnen würde, ist die Zuordnung falsch, so werden alle Gewichte des Netzwerkes anhand der *Lernregel* verändert. Diese maschinelle Lernmethode nennt sich *überwachtes Lernen* (engl. *supervised learning*) (Mallot, 2013, S 88).
Während Lernregeln früherer Netze, wie zum Beispiel die Hebb'sche Lernregel, noch biologisch inspiriert waren, wird heutzutage meist *Backpropagation* verwendet. Hier wird versucht den Fehlerterm E anhand der Veränderung des Gewichtes w_{ij} zu minimieren (Mallot, 2013, S 88):

$$\frac{\partial E}{\partial w_{ij}} = -o_i \cdot \delta_j.$$

Für ein Neuron der letzten Schicht ist

$$\delta_j = f'(\Sigma_i w_{ij} o_i - \theta_j) \cdot (t_j - o_j)$$

dabei ist t_j das Lehrersignal und gibt an, welches die richtige Antwort ist. Für Neurone in anderen Schichten wird über alle nachfolgenden Neurone k summiert:

$$\delta_j = f'(\Sigma_i w_{ij} o_i - \theta_j) \cdot (\Sigma_k \delta_k w_{jk}).$$

Hierbei handelt es sich um einen *stochastic gradient descent* (SGD). Das Verfahren steigt auf dem multi-dimensionalen Gradienten der Fehlerfunktion ab und lernt damit „gute“ Gewichte. Um die Ergebnisse für den Menschen nachvollziehbar zu machen, wird die Ausgabe der letzten Schicht meist durch die Softmax-Funktion

$$o_i = \frac{e^{o_i}}{\Sigma_k e^{o_k}}$$

gewichtet. Da die Summe aller Ausgaben dieser Schicht somit 1 ergibt, wird die Ausgabe für jede Klasse q_i meist als „Zuversicht des Netzwerkes“ oder „Wahrscheinlichkeit“ (engl. *probability*) bezeichnet.

Durch den Gradientenabstieg während des Trainings bilden sich rezeptive Felder. Erste Untersuchungen zeigen, dass die rezeptiven Felder auch ohne direkte Einschränkung den Anforderungen von Serre et al. (2007) entsprechen. Die rezeptiven Felder früher Schichten sind oftmals

Gabor Filter und die Komplexität der Filter steigt mit der Tiefe ihrer Schicht. Jedoch braucht es viele Iterationen und viele Trainingsdaten, bis sich sinnvolle rezeptive Felder bilden (Zeiler & Fergus, 2013; Yosinski, Clune, Nguyen, Fuchs & Lipson, 2015). „Eine Daumenregel besagt, dass ein überwachter Deep learning Algorithmus generell akzeptable Ergebnisse mit 5000 gelabelten Beispielen pro Kategorie erzielen wird und menschliche Leistungen erreicht oder übertrifft, wenn er mit einem Datensatz trainiert wird, der mindestens 10 Millionen gelabelte Beispiele beinhaltet “ (Goodfellow, Bengio & Courville, 2016, S. 20).

Ich möchte in dieser Arbeit die Klassifikationsleistung und die gelernten rezeptiven Felder eines tiefen Neuronalen Netzes untersuchen. Das Netzwerk soll lernen, Pflanzen zu unterscheiden. Diese Aufgabe ist aufgrund der geringen Unterschiede zwischen den Spezies und den großen Unterschieden innerhalb einer Spezies extrem schwierig.

In den folgenden Abschnitten möchte ich zuerst einen Einblick in den PlantCLEF Wettbewerb geben (Kapitel 3). Dieser Wettbewerb befasst sich mit der Klassifikation von Pflanzen anhand von Bildern. Die bisherigen Erkenntnisse des Wettbewerbs werden als Ausgangspunkt weiterer Untersuchungen verwendet. In Kapitel 4 werde ich auf das genutzte Netzwerk, sowie die Untersuchungsmethoden genauer eingehen. Dort finden sich auch weitere Exkurse zum Thema „tiefe Neuronale Netze“. Daraufhin werden in Kapitel 5 die Ergebnisse meiner Arbeit präsentiert und in Kapitel 6 erläutert und diskutiert.

3. PlantCLEF Challenge 2016

Das *LifeCLEF Lab* führt jährlich verschiedene Wettbewerbe aus, welche die Nutzung multimedialer Identifikationswerkzeuge, wie Smartphones oder Tablets, vorantreiben soll, um Wissen über die „Identität, geographische Verteilung und Entwicklung von Lebewesen“ zu erlangen (Joly et al., 2016). Einer dieser Wettbewerbe beschäftigt sich mit der Identifikation von verschiedenen Pflanzen anhand von Fotografien: die so genannte *PlantCLEF Challenge*. Die genaue Aufgabenstellung der jährlichen Wettbewerbe variiert leicht, um sich aktuellen Herausforderungen zu stellen und dem Ziel, ein optimales Identifikationswerkzeug zu erschaffen, anzunähern.

3.1. Aufgabe

Die spezifische Aufgabe des Jahres 2016 bestand in einer so genannten „open-world recognition“. Dies bedeutet, dass ein Erkennungswerkzeug nicht bekannte Klassen erfolgreich als *unbekannt* klassifizieren, aber gleichzeitig bekannte Pflanzenarten mit hoher Sicherheit und Präzision erkennen muss (Goëau, Bonnet & Joly, 2016). Diese Aufgabe ist besonders schwierig, da die Ausgaben von Neuronalen Netzen meist mit hoher Zuversicht getätigt werden, auch wenn sie falsch sind.
Der bereitgestellte Trainingsdatensatz enthält 1000 verschiedene Pflanzen-Spezies, welche der Klassifikationsapparat erkennen soll. Die Erkennungssoftware wurde dann auf einem Testdatensatz getestet, welcher die oben genannte Aufgabe überprüft. Mehr dazu folgt in den unteren Abschnitten *Datensatz* und *Metrik*.

3.2. Datensatz

Der Datensatz der PlantCLEF-Challenge des Jahres 2016 setzt sich aus einem Trainings-, sowie einem Testteil zusammen. Ersterer besteht aus 113205 Bildern und den zugehörigen Informationen (*siehe Auflistung S.9*). Jedes der Bilder des Trainingsdatensatzes gehört einer der 1000 Pflanzenarten an, welche das Netzwerk zu klassifizieren lernt. Der Testdatensatz wiederum besteht aus 8000 Bildern und den zugehörigen Informationen. 4633 dieser Bilder gehören zur Klasse der gelernten Spezies, die restlichen 3367 Bilder bilden entweder unbekannte Spezies oder nicht-pflanzliche Objekte ab. Es existieren Bilder für Kräuter, Bäume and Farne, die in

Frankreich und benachbarten Länder beheimatet sind (Goëau et al., 2016).
Die Bilder des Datensatzes werden von freiwilligen (Hobby-)Fotografen unter Angabe der unten abgebildeten Informationen eingereicht und anschließend von Mitgliedern der PlantCLEF Gemeinschaft bezüglich der Korrektheit und Qualität der Bilder bewertet. Falls das Bild genügend „Votes“ erreicht hat, wird es in den Datensatz aufgenommen.
Zu jedem Bild sind die folgenden Informationen gegeben, welche zur Klassifikation genutzt werden können:

- *ObservationId*: Referenznummer zur Identifikation einer spezifischen Pflanze
- *MediaId*: Referenznummer zur Identifikation des Bildes
- *Vote*: Anzahl der abgegebenen Stimmen zur Entscheidung, ob das Bild verwendet werden darf
- *Content*: Dargestelltes Bildorgan (Blatt, Blatt-Scan, ganze Pflanze, Stamm, Ast, Blüte, Frucht)
- *Author*: Fotograf des Bildes
- *ClassId*: Nummer zur Identifikation der Spezies einer Pflanze
- *Family*: Familie der abgebildeten Pflanze
- *Genus*: Gattung der abgebildeten Pflanze
- *Species*: Spezies der abgebildeten Pflanze
- *Date*: Aufnahmedatum des Bildes
- *Location*: Aufnahmeort des Bildes
- *Latitude*: Breitengrad des Aufnahmeortes
- *Longitude*: Längengrad des Aufnahmeortes
- *YearInCLEF*: Angabe, seit welchem Jahr sich das Bild im Datensatz befindet
- *ObservationId2014*: Angabe der alten ObservationId, falls das Bild sich 2014 bereits im Datensatz befand
- *ImageId2014*: Angabe der alten ImageId, falls das Bild sich 2014 bereits im Datensatz befand
- *LearnTag*: Zugehörigkeit zum Testset oder Trainingsset

Für den Testdatensatz sind die Angaben zur „ClassId“, „Family“, „Genus“ und „Species“ nicht gegeben.

In Abbildung 3.1 ist die Verteilung der Bildanzahl pro Klasse zu sehen. Man erkennt, dass manche Klassen sehr stark und andere nur wenig vertreten sind. Dies fällt besonders bei der Familie der *Asteraceae* auf, für die ganze 8313 Bilder vorhanden sind.

Die 124 im Datensatz enthaltenen Familien spalten sich in 516 Gattungen und diese wiederum in 1000 Spezies auf.
Der Trainingsdatensatz setzt sich aus 13367 Bildern von Blättern, 12605 Bildern von abgescannten Blättern, 16236 Bildern der ganzen Pflanze, 5476 Bildern des Stammes, 8130 Bildern von Ästen, 28225 Bildern von Blüten und 7720 Bildern von Früchten zusammen.

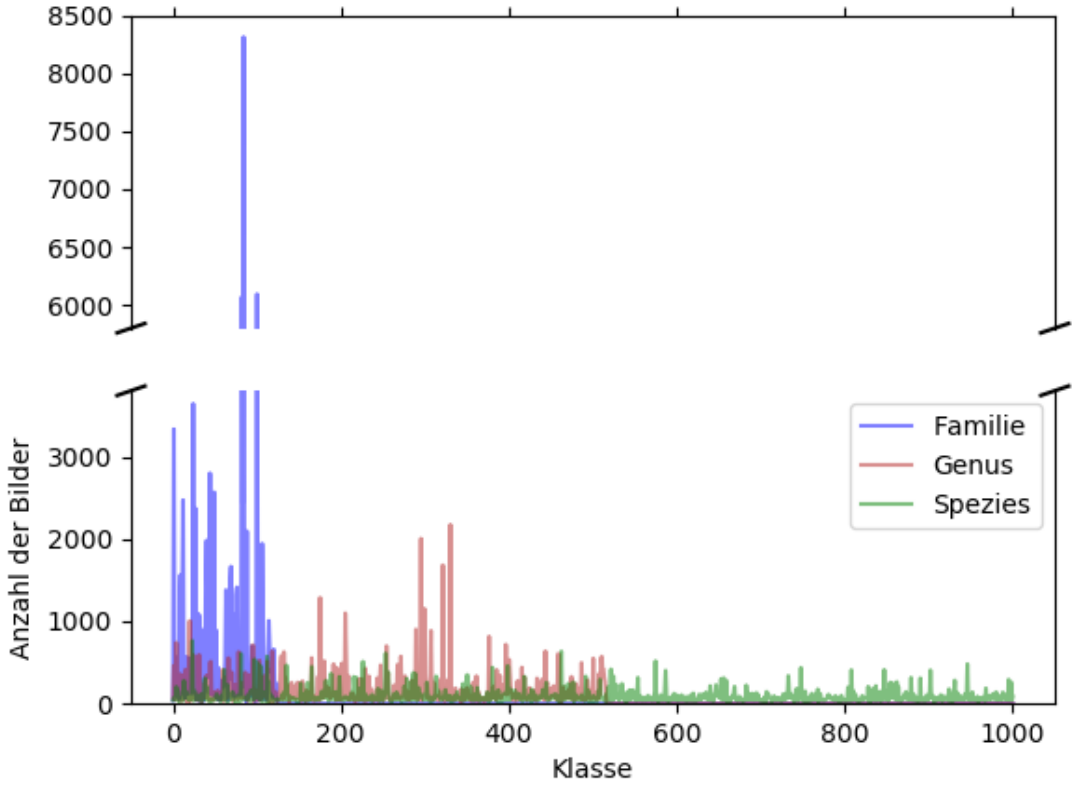

Abbildung 3.1.: Anzahl der Bilder pro Klasse

3.3. Metrik

Die Organisatoren der PlantClef-Challenge nutzten die „mean average precision“ oder kurz *mAP* um die *open-world-recognition* zu testen.
Für jede Klasse q_i wurden alle Vorhersagen der Einreichung (engl. *prediction*) ihrem Softmax-output (engl. *propbability*) nach geordnet. Nun wurde für diese Vorhersagen jeder Klasse q_i die *average precision* (AP) berechnet und anschließend über alle Klassen Q gemittelt.

$$mAP = \frac{\Sigma_{q=1}^{Q} AP(q)}{Q}$$
$$AP = \Sigma_{k=1}^{n} P(k) \triangle r(k)$$

k ist der Rang in der geordneten Liste der Vorhersagen, n die Anzahl der Klassen. $P(k)$ ist die Präzision bis zur k-ten Stelle und $\triangle r(k)$ ist die Veränderung des „Recalls“ zwischen $k-1$ und k [1].

Die Metrik wurde für drei verschiedene Testfälle genutzt. Der **mAP-open**-Score gibt die mean average precision für 1000 Spezies auf dem gesamten Testdatensatz an. Der **mAP-closed**-Score wiederum gibt den mAP Score auf dem Testdatensatz ohne Distraktoren an. Solche Distraktoren sind die nicht-bekannten und nicht-pflanzlichen Objekte. Der **mAP-open-invasive** Score gibt die Präzision für 26 ausgewählte Spezies im Testdatensatz an.
Um die *open-world-recognition* zu testen, ist somit der **mAP-open**-Score am aussagekräftigsten.

3.4. Ergebnisse der PlantCLEF Challenge 2016

Acht Gruppen reichten ihre Erkennungssysteme ein. Jede der Gruppen konnte bis zu vier verschiedene „runfiles“ einbringen. Diese enthielten die Voraussagen ihrer Erkennungssysteme für jedes im Testdatensatz enthaltene Bild in folgendem Format: <*MediaID;Vorausgesagte Klasse;Sicherheit*>.
Die besten Einreichungen stammten von den Gruppen „Bluefield“ aus Japan, „Sabanci“ aus der Türkei und „CMP“ aus der Tschechischen Republik. Ihre Systeme erreichten einen maximalen mAP-closed von 74.2% (Bluefield), 73.8% (Sabanci) beziehungsweise 71.0% (CMP) (Hang, Tatsuma & Aono, 2016; Ghazi, Yanikoglu & Aptoula, 2016; Šulc, Mishkin & Matas, 2016). [2]

[1] `https://en.wikipedia.org/wiki/Information_retrieval`, 06.08.2017
[2] Die Einordnung dieser Resultate wird in Abschnitt A.2 im Anhang diskutiert.

4. Methodik

Um die rezeptiven Felder eines Neuronalen Netzes zu untersuchen, musste ein Netz mit sinnvollen und an die Pflanzenerkennung angepassten rezeptiven Feldern generiert werden. Hierfür wurde ein Convolutional Neural Network aus dem PlantCLEF Wettbewerb des Jahres 2016 repliziert und darauf aufbauend Untersuchungen am Netzwerk durchgeführt. Ein Convolutional Neural Network nutzt Faltungen, anstatt voll verbundener Schichten. Hierdurch können die entstandenen rezeptiven Filter auf dem ganzen Bild angewendet werden. Nähere Informationen hierzu folgen in Abschnitt 4.3.2.
In diesem Kapitel möchte ich in Abschnitt 4.1 und Abschnitt 4.2 die verwendeten Rechnerarchitektur und Programme vorstellen, mit denen die Netzwerke trainiert, getestet und untersucht wurden. Daraufhin wird in Abschnitt 4.3 auf das replizierte Netzwerk eingegangen, wobei die verwendete Architektur, die Preprocessing- und Trainingsmethoden genauer erklärt werden. In Abschnitt 4.4 möchte ich letztendlich die Methoden verstellen, welche ich zur Untersuchung des Netzwerkes nutzte.

4.1. Rechnerarchitektur

Die Netzwerke wurden auf einem Rechner mit den folgenden Eigenschaften trainiert:

- *Prozessor*
  - Hersteller: Intel
  - Name: Intel(R) Core(TM) i5-6500 CPU
  - Prozessorkernzahl: 4
  - Grundtaktfrequenz: 3.20GHz
- *Graphikkarte* (GPU)
  - Hersteller: NVIDIA
  - Name: GeForce GTX 1050 Ti
  - Speichergröße: 4GB
  - Cuda-Recheneinheiten: 768

Darüber hinaus stand der Baden-Württemberg Cluster zur Untersuchung des Netzwerkes zur Verfügung. Dessen GPUs sind Tesla K80 der Marke NVIDIA mit einer Speichergröße von 12GB.

4.2. Tensorflow und Tensorflow-Slim

Um die Neuronalen Netzwerke aufzubauen und zu trainieren wurde Tensorflow verwendet, welches die wichtigsten und bekanntesten Algorithmen und Methoden implementiert hat und ein schnelles Training auf GPUs erlaubt.
Tensorflow wurde vom Google Brain Team [1] entwickelt und ist ein Open Source framework zur numerischen Berechnung. Dabei nutzt es Flussdiagramme in welchen jede mathematische Operation durch einen Knoten dargestellt wird. Die Kanten, welche die Knoten verbinden, sind Tensoren, also multidimensionale Daten-Arrays. Tensorflow's API ist in verschiedenen Programmiersprachen verfügbar, jedoch am besten für Python dokumentiert. Durch die flexible Architektur ist es möglich, mehrere GPUs oder CPUs zu nutzen, um die Dauer des Trainings eines Neuronalen Netzes zu verringern.
Tensorflow-Slim wurde von Guadarrama und Silberman [2] entwickelt und baut auf Tensorflow auf. Es erlaubt es auf einfache Art und Weise Neuronale Netze zu definieren, aufzubauen und zu evaluieren. Dabei nutzt es die Methoden und Funktionen von Tensorflow und verbindet diese. Aus diesem Grund sind Tensorflow und Tensorflow-Slim voll kompatibel. Der größte Vorteil dieser API sind jedoch die auf dem ImageNet Datensatz von 2012 vortrainierten Netzwerke. Für diese Netzwerke sind die Architektur, sowie die Gewichte downloadbar [3]. So zum Beispiel das ResNet (He, Zhang, Ren & Sun, 2015), das Inception Netzwerk (Szegedy et al., 2014) und das VGG Netz (Simonyan & Zisserman, 2015). Diese Netzwerke können dann in ihrem Aufbau verändert, direkt zur Klassifikation von bekannten Klassen genutzt oder zur Bestimmung bisher unbekannter Klassen trainiert werden. Letzterer Prozess nennt sich *finetuning*.

4.3. Replikation CMP paper

Als Grundlage für weitere Untersuchungen wurden die Netzwerkarchitektur, die Bildvorbearbeitung (*preprocessing*) und die Hyperparameter der *CMP* Gruppe (Šulc et al., 2016) herangezogen, welche bei der PlantCLEF challenge 2016 den dritten Platz belegte (Goëau et al., 2016). Hyperparameter sind Werte die vor dem Start des Trainings manuell festgelegt werden müssen. So zum Beispiel die Lernrate.
In den folgenden Abschnitten werden zuerst der Ansatz der CMP Gruppe und daraufhin meine

[1] `https://www.tensorflow.org/`, 07.08.2017
[2] `https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim`, 07.08.2017
[3] `https://github.com/tensorflow/models/tree/master/slim`, 07.08.2017

durchgeführten Umsetzung aufgeführt. Zudem werden kurze Einführungen in die Methoden gegeben und deren Vorteile erläutert.

4.3.1. Preprocessing

Wenn einem Menschen ein ihm unbekanntes Objekt präsentiert wird, welches er sich merken soll, wird er es aus verschiedenen Blickwinkeln und wenn möglich mit verschiedenen Sinnen betrachten, um möglichst viele Informationen über den Gegenstand zu sammeln. Ein Neuronales Netzwerk hat meist nicht die Möglichkeit verschiedene Sinne zu nutzen oder ein Objekt selbst zu inspizieren. Es ist darauf angewiesen die statistischen Eigenschaften eines Bildes zu verwenden, um diese soweit zu verarbeiten, bis es zu einer Ausgabe gelangt. Dabei spielen in ersten Schritten zum Beispiel die Ähnlichkeit benachbarter Pixel eine Rolle, da diese Kanten bedeuten könnten. In höheren Abstraktionsstufen werden auch komplexere Muster verwendet, so könnte sich beispielsweise ein Detektor für Pflanzenblätter bilden (Zeiler & Fergus, 2013). Ein solches Blatt könnte allerdings mit dem Stiel nach oben, nach unten oder zur Seite und zudem in vielen verschiedenen Größen und an verschiedenen Bildpositionen. Diese Veränderungen der Bilder kann das künstliche Netz für seine Erforschung des Gegenstandes nutzen. Durch das Preprocessing werden diese modifizierten Bilder hergestellt.
Des Weiteren wird in diesem Verarbeitungsschritt dafür gesorgt, dass die Bilder die richtige Größe besitzen und ihre Pixelwerten werden standardisiert. Das Abziehen des Mittelwertes zentriert die Datenpunktewolke um den Nullpunkt. Außerdem kann eine Normalisierung der Daten durchgeführt werden, entweder durch eine Skalierung mit der Standardabweichung jeder Dimension oder durch Skalierung der Dimensionen auf Werte zwischen -1 und 1.
Die daraus resultierenden Bilder führen zu verbesserten Lernleistungen.

Das *Preprocessing* des CMP-Netzwerkes war unterteilt in die Verarbeitung für den Trainings- und die Verarbeitung für den Testprozess. Während des Trainings wurden alle Bilder auf die nötige Größe skaliert und anschließend horizontal gespiegelt. Sowohl das skalierte Originalbild, als auch das skalierte und gespiegelte Bild wurden für das Training genutzt. Weitere Preprocessing Schritte wurden nicht durchgeführt.
Während der Evaluierung wurden die Originalbilder auf ihre 4 Ecken und einen zentralen Ausschnitt zugeschnitten. Die resultierenden Bilder wurden daraufhin horizontal gespiegelt. Die Voraussagen für diese 10 Bilder wurden gemittelt.

Mein *preprocessing* für das **Training** setzte sich für die verschiedenen Netzwerke wie folgt zusammen:

1. Netzwerk 1
    - Skalieren und zufälliges zuschneiden des Bildes auf die benötigte Größe von 224 x 224 Pixeln

- Horizontales Spiegeln der Bilder
- Zufällige Variation der Farbwerte

2. Netzwerk 2: **Ab hier wurde keine horizontale Spiegelung mehr vorgenommen, da unserer Meinung nach durch Spiegelungen Tiefeinformationen verloren gehen kann**
   - Skalieren und zufälliges zuschneiden des Bildes auf die benötigte Größe von 224 x 224 Pixeln
   - zufällige Variation der Farbwerte
3. Netzwerk 3
   - Skalieren und zufälliges zuschneiden des Bildes auf die benötigte Größe von 224 x 224 Pixeln
4. Netzwerk 4, Netzwerk 5: **Ab hier wurden Fließkommazahlen im Intervall [-1,1] anstatt der Ganzzahlen im Intervall [0, 255] verwendet**
   - Skalieren des Bildes auf die benötigte Größe von 224 x 224 Pixeln
   - Skalieren der Pixelwerte auf ein Intervall von [-1, 1]
5. Netzwerk 6, Netzwerk 7, Netzwerk 8
   - Skalieren des Bildes auf die benötigte Größe von 224 x 224 Pixeln
   - Abzug des bildinternen Pixelmittelwertes
   - zufällige Drehung des Bildes um 0°, 90°, 180°oder 270°

Für die **Evaluation** wurde folgendes perprocessing verwendet:

1. Netzwerk 4, Netzwerk 5
   - Bildausschnitte
     - alle 4 Ecken
     - ein zentraler Bildausschnitt
     - das Gesamtbild
   - Skalieren der Bilder auf die benötigte Größe
   - Skalieren der Pixelwerte auf ein Intervall von [-1, 1]
2. Netzwerk 6, Netzwerk 7, Netzwerk 8
   - Ausschneiden folgender Bildausschnitte
     - alle 4 Ecken
     - ein zentraler Bildausschnitt
     - das Gesamtbild

- Skalieren der Bilder auf die richtige Größe
- Abzug des bildinternen Mittelwerts der Pixel

4.3.2. Architektur

Bereits 1998 nutzen LeCun, Bottou, Bengio und Haffner (1998) ein Convolutional Neural Network, um handgeschriebene Buchstaben zu erkennen und spätestens seit *AlexNet* von Krizhevsky, Sutskever und Hinton (2012) sind Convolutional Neural Networks aus der Bilderkennung nicht mehr wegzudenken. Die Anzahl der Schichten wird dabei von Jahr zu Jahr größer, da die Rechenleistung moderner Grafikkarten dies erlaubt und die Klassifikationsleistungen hierdurch meist erhöht werden.
Das CMP Team nutzte das bekannte ResNet-152 von He et al. (2015). Dieses 152-schichtige Netzwerk basiert auf hintereinander geschalteten „*bottlenecks* ". Diese Engpässe helfen dabei das Problem der *vanishing gradients* (Abschnitt 4.1) zu beheben und erlauben somit ein Training von viel tieferen Netzen.
Um die Generalisierungsleistung zu erhöhen, wurde außerdem eine sogenannte „*Maxout*"-Schicht (Goodfellow, Warde-Falrey, Mirza, Courville & Bengio, 2013) mit vier linearen Teilstücken verwendet, jedes Teilstück wiederum bestand aus 512 Neuronen. Dabei schaut jede lineare Schicht auf den gleichen Input und die nachgeschaltete Maxout-Funktion wählt für jedes Neuron das am stärkste aktive aus (Abschnitt 4.1). Diese Regularisierungsform ist besonders effektiv, wenn sie mit Dropout (Srivastava, Hinton, Krizhevsky, Sutskever & Salakhutdinov, 2014) gepaart wird, weshalb dieses der Maxout-Schicht nachgeschaltet ist (Abschnitt 4.1).

Aufgrund von Hardware- und Trainingszeiteinschränkungen nutzte ich das 50 Schichten tiefe ResNet-50, welches laut dem offiziellen Paper von He et al. (2015) bei der ImageNet-challenge einen Top-5 Fehler von 6.71 % und einen Top-1 Fehler von 22.85% hat. Im Vergleich dazu sinken die Fehler des ResNet-152 auf gerade einmal 5.71% beziehungsweise 21.43%.
Darauf aufbauend wurde, wie oben beschrieben, eine Dropout und eine Maxout-Schicht aufgesetzt, bevor ein voll verbundener (*fully connected*) Softmax-Klassifikator das Netzwerk abschloss.

Exkurs: Convolution

Es ist nicht sinnvoll hoch dimensionale Eingaben voll mit der nächsten Schicht zu verbinden, da hier zu viele Gewichte gelernt und gespeichert werden müssen. Stattdessen betrachtet man ein lokales Feld (Breite x Höhe) in der ganzen Tiefe der Eingabe (z.B. Tiefe 3, falls es sich um ein RGB Bild handelt). Dies ist das so genannte *rezeptive Feld* des Neurons. Hat das Netzwerk ein sinnvolles rezeptives Feld (*Filter*) entwickelt, so ist es nötig dieses auf dem ganzen Bild nutzen zu können und nicht nur an einer Stelle des

Inputs. Deshalb lässt man diesen Filter über das ganze Bild wandern und erhält dann die so genannte *feature map* als Ausgabe. Die *feature map* gibt das Resultat des Filters an jeder betrachteten Stelle des Bildes an.
Während einer Faltung (*convolution*) kann man 3 Parameter frei wählen. Erstens die Größe des Filters (Höhe x Breite) und die Tiefe, also die Anzahl der Filter. Des weiteren den *stride*. Der stride gibt die Schrittgröße an in welcher der Filter über das Bild wandert. Zuletzt kann die Eingabe noch in ein *zero-padding* eingebettet werden, sprich um das Bild herum werden n Reihen von Nullen eingefügt. Dies erlaubt die Größe der Ausgabe zu beeinflussen.
In Abbildung 4.1 ist eine Faltung beispielhaft dargestellt. Die Eingabe der Größe 6 x 6 (grau) wird durch den 2 x 2 Filter (rot) gefaltet. Dabei schreitet der Filter mit *stride* 2 über die Eingabe. Daraus resultiert die 3 x 3 *feature map* (blau). Diese Ausgabe wird nun als Eingabe der nächsten Schicht weitergegeben.
1 x 1 Convolutions wurden erstmals von (Lin, Chen & Yan, 2014) verwendet. Auf der räumlichen Ebene führen sie keine Veränderung hervor, jedoch arbeiten sie auf der ganzen Tiefe der Eingabe. Deshalb können sie auch anstelle von voll verbundenen Schichten verwendet werden.

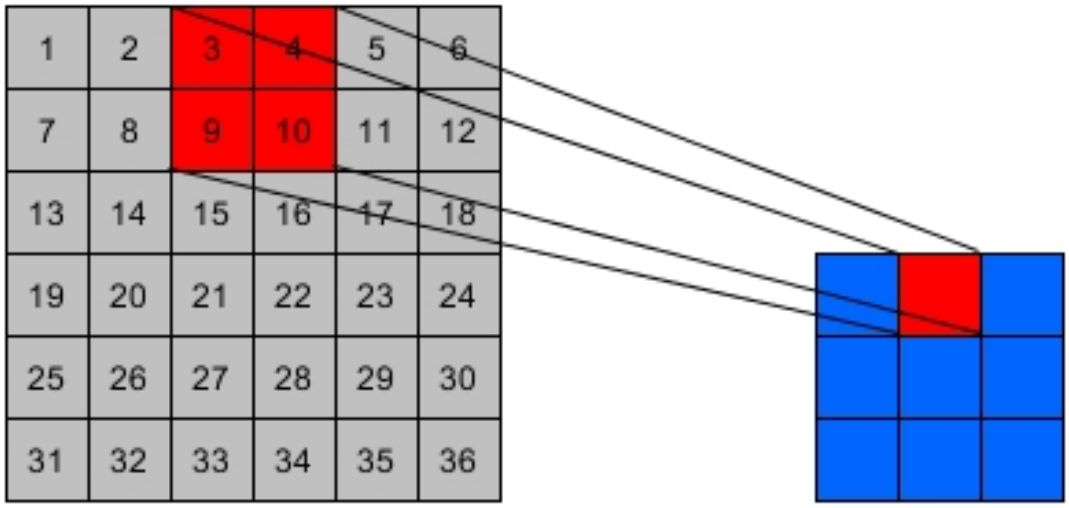

Abbildung 4.1.: *Links*: Faltung der 6x6 Eingabe (grau) durch den 2x2 Filter (rot) mit Stride 2. *Rechts*: Enstandene 3x3 feature map (blau), mit dem gerade aktiven feature in rot.

Exkurs: Vanishing Gradients

Beim Training tiefer Neuronaler Netze wird der Fehler vom hinteren Ende zum vorderen Ende des Netzes propagiert. Dabei flacht der Gradient immer weiter ab, wodurch Schichten nahe der Eingabe nur noch geringe Gewichtsveränderungen erfahren. Da diese frühen

Schichten somit fast keinen Lernfortschritt erzielen, müssen spätere Schichten auf sehr verrauschten oder sogar zufälligen Eingaben weiterarbeiten. Dies hindert das Netzwerk daran sinnvolle Ausgaben zu liefern.
Die Lösung dieses Problems ist äußerst schwierig, wird aber durch das Residuale Netzwerk gelöst. Der Gradient kann hier aufgrund sogenannter „shortcut"-Verbindungen ohne Verluste durch das Netzwerk propagiert werden.

Exkurs: Pooling

Pooling reduziert die räumliche Dimension der Eingabe, indem die Pooling-Operation über den Input wandert. Dabei wird je ein Ausschnitt der Größe h x w betrachtet. Aus jedem Bildausschnitt wird nun zum Beispiel das Maximum oder der Durchschnitt ausgewählt und als Ausgabe dieses räumlichen Punktes gewählt. Ähnlich wie bei der Faltung kann auch hier die Größe (Breite x Höhe) des zu betrachtenden Bildausschnittes, sowie die Schrittgröße (stride) gewählt werden. Allerdings betrachtet eine Pooling-Operation nur die räumlichen Dimensionen der Eingabe und nicht die Tiefe.
Durch die räumliche Dimensionsreduktion müssen weniger Parameter trainiert werden, was die Berechnungszeit verbessert und zu einer besseren Generalisierung führt. Dabei werden Statistiken der Bildeigenschaften pro Bildregion hergestellt, anstatt die Aktivität aller rezeptiven Felder an jeder Stelle des Bildes zu betrachten (Goodfellow et al., 2016, S.339ff).

Exkurs: Dropout

Ein Trainingsdatensatz enthält Informationen zur Abbildung zwischen Eingabe und Ausgabe. Aber er enthält auch Fehler aufgrund der zufälligen Stichprobe (*sampling error*). Ein Netzwerk welches diesen *sampling error* lernt und seine Voraussagen aufgrund dieser Eigenschaften macht, wird auf einem neuen Datensatz nur schlechte Ergebnisse liefern. Dieser Vorgang nennt sich *overfitting*.
Bei Dropout (Hinton, Srivastava, Krizhevsky, Sutskever & Salakhutdinov, 2012; Srivastava et al., 2014) wird bei jedem Trainingsschritt nur eine zufällige Untermenge der Größe $\frac{1}{n}$ aller Neuronen verwendet. Die Ausgabe der nicht verwendeten Neuronen wird auf 0 gesetzt und die Ausgabe der verwendeten Neurone wird mit n multipliziert.
Während der Evaluation werden alle Neuronen verwendet, ihre Ausgabe wird nicht skaliert. Hierdurch wird sogenanntes *model averaging* durchgeführt. Sprich in einem Netzwerk entstehen mehrere voneinander unabhängige Netzwerke, welche dann zusammen eine Voraussage abgeben.
Hinton et al. (2012) erklären, dass es auf diese Weise möglich ist ein Netzwerk vor overfitting zu bewahren, da ein Neuron sich nicht darauf verlassen kann, dass andere Neuronen in jedem Schritt aktiv sind. Es muss unabhängige *Merkmalsdetektoren* entwickeln, welche auch ohne weitere Informationen hilfreich sind.

Dropout gehört somit zu den sogenannten Regularisierungsmethoden. Diese Regularisierungsmethoden sorgen dafür, dass ein Netzwerk auch für unbekannte Daten gute Resultate erzielt.

Exkurs: Max-Out

MaxOut ist eine Aktivierungsfunktion, welche von Goodfellow et al. im Jahr 2013 vorgestellt wurde. Es besteht aus k linearen Teilstücken aus welchen das jeweilige Maximum berechnet wird (siehe Abbildung 4.2):

$$\begin{aligned} h_i(x) &= max_{j\in[1,k]} z_{ij} \\ &= max_{j\in[1,k]} x^T W_{i,j} + b_{i,j} \end{aligned}$$

Dabei sei *x* der Input, *i* und *j* die miteinander verbundenen Neurone und *k* die Anzahl der linearen Teilstücke.
Diese Aktivierungsfunktion erlaubt es konvexe Funktionen besser zu approximieren und ergibt verbunden mit Dropout bessere *model averaging* Resultate. Dies liegt laut den Autoren daran, dass sigmoide Aktivierungsfunktionen kurvenartige Verläufe haben und diese von Dropout schlechter gelernt werden. Die linearen Operationen gepaart mit der Maximum-Funktion seien für Dropout besser geeignet.

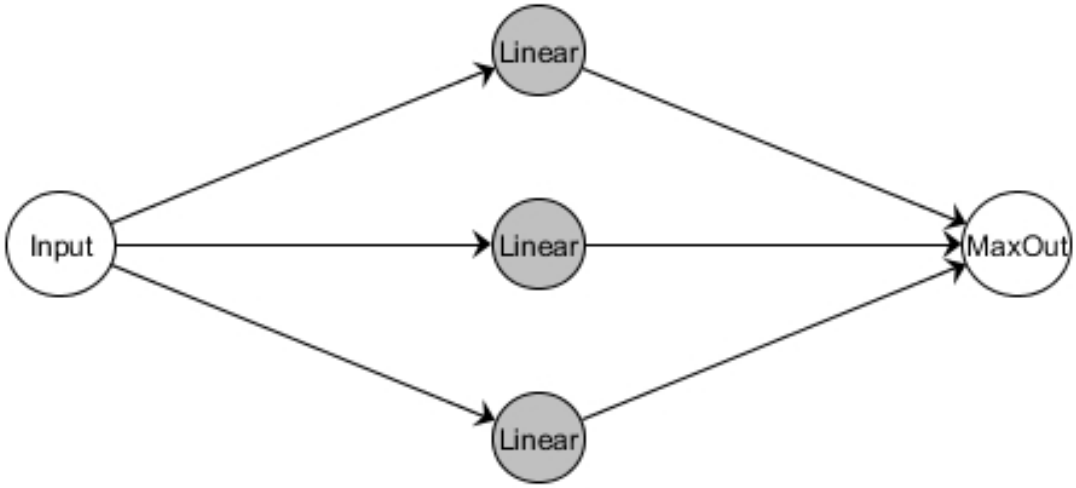

Abbildung 4.2.: MaxOut mit 3 linearen Teilstücken (grau). Dargestellt für eine Schicht und je ein Input- und Outputneuron (weiß).

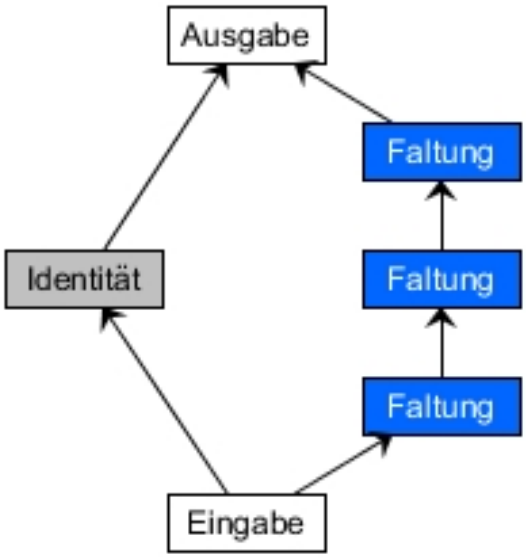

Abbildung 4.3.: Flaschenhals-Modul zur Berechnung des Residuals. Die Identitätsfunktion (grau) verbindet die Eingabe mit der Ausgabe genauso, wie die hintereinander ausgeführten Faltungen.

ResNet

Das tiefe residuale Netzwerk, kurz ResNet, baut sich aus hintereinander geschalteten *bottlenecks* auf (siehe Abbildung 4.3). Jeder dieser Engpässe lernt die Funktion der Unterschiede zwischen zwei Schichten (Residualen) $\mathcal{F}(x)$, anstatt der eigentlich gewollten Funktion $\mathcal{H}(x)$. Der Flaschenhals besteht aus drei nicht linearen Faltungsoperationen (*convolutional layer*) und einem linearen Byepass, dem sogenannten „shortcut“. Je eine Gruppe von bottlenecks wird zu einem *Block* zusammengefasst, dabei ist der erste shortcut eines Blockes eine Convolution, um die Anzahl der rezeptiven Felder zu erhöhen. Der letzte shortcut ist ein Max-Pooling, um die Dimensionen anzupassen. Die restlichen Verbindungen nutzen die Identitätsfunktion.
Das Netzwerk lernt somit $\mathcal{F}(x) := \mathcal{H}(x) - x$ und man erreicht die ursprünglich gewünschte Abbildung durch das Hinzufügen des Inputs x: $\mathcal{H}(x) = \mathcal{F}(x) + x$. Meine genutzte Netzarchitektur setzt sich wie folgt zusammen (siehe auch Abbildung 4.4):

1. Input Bild (224 x 224 x 3)(Breite x Höhe x RGB-Channel)
2. 7 x 7 Faltung mit Stride 2
3. 3 x 3 Max-Pooling mit Stride 2
4. Block 1
   a) Unit 1 / *bottleneck*
      - Convolution 1: 1 x 1, Stride 1
      - Convolution 2: 1 x 1, Stride 1
      - Convolution 3: 1 x 1, Stride 2
      - Shortcut: Convolution 1 x 1, Stride 1
   b) Unit 2 / *bottleneck*
      - Convolution 1: 1 x 1, Stride 1
      - Convolution 2: 1 x 1, Stride 1
      - Convolution 3: 1 x 1, Stride 2
      - Shortcut: Identity
   c) Unit 3 / *bottleneck*
      - Convolution 1: 1 x 1, Stride 1
      - Convolution 2: 1 x 1, Stride 1
      - Convolution 3: 1 x 1, Stride 2
      - Shortcut: Maxpooling 2 x 2, Stride 1
5. Block 2
   - 4 *bottlenecks*
6. Block 3
   - 6 *bottlenecks*
7. Block 4
   - 3 *bottlenecks*
8. Batch-Normalisierung
9. Pooling anhand des *Durchschnitts*
10. Max-Out mit 4 Linearen Teilstücken
11. Fully-Connected mit 1000 Neuronen und Softmax-Aktivierung

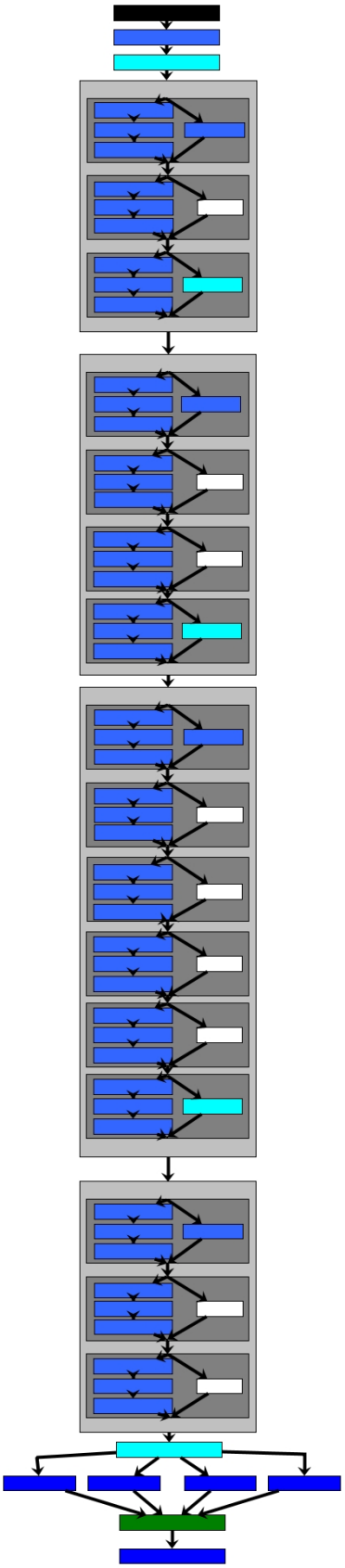

Abbildung 4.4.: Der Aufbau meines Neuronalen Netzes: Input (schwarz), Convolution (hellblau), Max-Pooling (türkis), Fully Connected (dunkelblau), Identität (weiß), Block (hellgrau), bottleneck (dunkelgrau), MaxOut (grün)

4.3.3. Training

Wie in der Einleitung bereits erklärt, ist die Größe des Datensatzes ausschlaggebend für die Leistung eines Neuronalen Netzes. Mit nur 113205 Bildern ist es nicht möglich, dein tiefes Neuronales Netz von Grund auf zu trainieren. Deshalb wurde ein auf dem ImageNet 2012 Datensatz „vortrainiertes“ Netz „finegetuned“. [4] Das bedeutet, dass die Gewichte jeder einzelnen Schicht nicht zufällig initialisiert sind, sondern von einem Netzwerk übernommen werden, welches auf anderen Bildern und Labeln trainiert war. Daraufhin werden die letzten Schichten abgeschnitten und zufällig initialisiert. Zuletzt werden alle Schichten des Netzes normal trainiert.
Um das Netzwerk zu trainieren müssen einige Hyperparameter und Optimierungsmethoden sinnvoll gewählt werden, um bereits entstandene rezeptive Felder nicht zu zerstören. Šulc et al. (2016) trainierten ihr erfolgreichstes Netzwerk für 150000 Iterationen mit einer Lernrate λ von 10^{-3}. Nach 100000 Iterationen wurde λ auf 10^{-4} heruntergesetzt. Als Gradientenabstiegsmethode wurde das *Momentum update* verwendet (Abschnitt 4.3.3). Dabei fließen in die Anpassung der Gewichte des Neuronalen Netzes sowohl die aktuellen Gradienten, als auch die bisherige allgemeine Richtung ein. Der Momemtum Faktor, welcher bestimmt wie stark die vorherigen Schritte mit einbezogen werden, wurde auf 0.9 gesetzt. Um die Gewichte davor zu bewahren zu groß zu werden und sich somit zu stark auf die Trainingsmuster und zu wenig auf unbekannte Daten zu konzentrieren, wurde ein *weight decay* von $2 \cdot 10^{-4}$ verwendet (Abschnitt 4.3.3). Dabei werden die Gewichte des Netzwerks in jedem Schritt um diesen weight decay Faktor verringert. Pro Trainingsschritt wurden 28 Bilder in einem *batch* präsentiert (Abschnitt 4.3.3). Die Verbindung mehrerer unterschiedlicher Netzwerke zur Klassifikation eines Bildes erhöht in den meisten Fällen die Präzision. Das CMP Team nutzte deshalb auch ein solches *ensemble* oder *bagging* von drei Netzwerken. Dabei wurde der Trainingsdatensatz in drei gleich große Teile aufgeteilt und jedes Netzwerk auf zwei dieser drei Teildatensätzen trainiert und auf dem anderen validiert (Abschnitt 4.3.3). Im Ganzen hat das Ensemble der Netzwerke also alle Trainingsdaten gesehen.

Alle meine Netzwerke[5] wurden für 150000 Iterationen mit den oben aufgeführten Hyperparametern und Optimierungsmethoden trainiert. Des Weiteren wurden für *Netzwerk 1* und *Netzwerk 2* andere Lernraten getestet:

	Netzwerk 1					**Netzwerk 2**	
Schritte	< 20000	< 35000	< 40000	< 55000	> 55000	< 100000	> 100000
Lernrate	0.001	0.0005	0.001	0.0005	0.0001	10^{-4}	10^{-5}

[4] `http://www.image-net.org/challenges/LSVRC/2012/`, 14.08.2017
[5] Mit Außnahme von Netzwerk 8

Exkurs: Batching

Während des Trainings können die Gewichte eines Netzwerks zu verschiedenen Zeitpunkten angepasst werden. Die am einfachsten zu berechnende Methode ist das Update nach der Präsentation eines Bildes. Dieses *Online*-Update bringt jedoch schlechte und nur langsame Konvergenz, da sich die Fehler und das Rauschen der einzelnen Eingaben nicht ausgleichen können. Das andere Extrem ist das Update nach der Präsentation *aller* Bilder. Dieses Verfahren bringt zwar die beste Gewichtsveränderung, ist in der Praxis allerdings nicht für große Datensätze berechenbar. Eine Mischung aus den beiden Verfahren ist das *batching*. Hierbei werden die Gewichte nach n Schritten verändert. n kann frei gewählt werden. Um die Konvergenz zu verbessern, sollte n jedoch auf einen möglichst großen Wert gesetzt werden.

Exkurs: Momentum

Momentum ist eine Erweiterung des klassischen Backpropagation Algorithmus, der in der Einleitung präsentiert wurde. Dabei wird ein Gewicht nicht nur anhand des momentanen Gradienten verändert, sondern auch anhand der letzten Schritte. Hierzu wird der letzte Update-Schritt gespeichert, gewichtet und zum momentanen Gradienten addiert:

$$\nabla_p w_{ij}(t+1) = \eta \cdot o_{pi} \cdot \delta_{pj} + \alpha \nabla_p w_{ij}(t).$$

Dabei sei *p* ein pattern (Eingabe), *t* die Zeit, *i* und *j* die miteinander verbundenen Neurone und η die Lernrate. δ ist der zurückpropagierte Fehlerwert.
Momentum führt zu einer schnelleren Konvergenz der Netzwerke, da die Fehlerfunktionen tiefer Neuronaler Netze in der Nähe von Minima oft sehr flach sind und ein klassischer Gradientenabstieg deshalb auch nur kleine Gewichtsaktualisierungen durchführt. Durch den Momentum Term können größere Schritte erreicht werden, falls der Gradient in jedem Zeitpunkt in die gleiche Richtung zeigt. Falls sich die Gradienten widersprechen, lenkt das Momentum den Abstieg in die Richtung, die für die präsentierten Eingaben im Mittel optimal ist.

Exkurs: Weight decay

Weight decay ist eine weitere Möglichkeit zur Regularisierung des Netzwerkes während des Trainings. Bei jedem Schritt in dem die Gewichte des Netzwerkes verändert werden, wird ein kleiner Teil der alten Gewichtsstärke abgezogen. Dadurch wird die Initialisierung

der Gewichte weniger wichtig und die Generalisierungsleistung steigt.
Dabei verändert sich der Aktualisierungsschritt wie folgt:

$$\nabla_p w_{ij}(t+1) = \eta \cdot o_{pi} \cdot \delta_{pj} - d \cdot w_{ij}(t).$$

Der weight decay Faktor d wird hierbei meist auf Werte zwischen 0.005 und 0.03 gesetzt.

Exkurs: Trainings-, Validierungs- und Testdatensatz

Während des Trainings eines Maschinellen Lernsystems werden meist drei Arten von Datensätzen genutzt. Das System wird auf einem Trainingsdatensatz trainiert, dieser enthält die Daten, welche gelernt werden. Nur anhand dieser Daten werden z.B. die Gewichte angepasst. Der Validierungsdatensatz wird meist vor dem Training vom Trainingsdatensatz ausgegliedert. Er wird genutzt um während des Trainings immer wieder zu testen, ob das System gut generalisiert, also auch auf unbekannten Daten gute Vorhersagen gibt. Der Testdatensatz wiederum wird dem System erst nach dem Training präsentiert. Er testet wie gut das System wirklich auf unbekannten Daten arbeitet.

Exkurs: Ensemble / Bagging

In der Psychologie beschreibt der Begriff „*wisdom of the crowd* “ die Eigenschaft, dass Rateversuche näher an tatsächlichen Ergebnissen liegen, wenn sie über mehrere Personen gemittelt sind. Dies liegt daran, dass sich verschiedene Biases ausgleichen. Ein ähnliches Phänomen ergibt sich bei Neuronalen Netzen. Wird die Voraussage (prediction) mehrerer Netzwerke gemittelt, so ist diese Ausgabe besser als die Voraussagen der einzelnen Netzwerke. Speziell beim oben beschriebenen open-world-recognition task ergibt es Sinn ein Ensemble zu nutzen, da sich auf diese Weise fehlerhafte Voraussagen relativieren und nur übereinstimmende Voraussagen eine hohe Aussagekraft haben.
Die Verbindung mit *cross-validation* ist eine Form des Baggings. Der Trainingsdatensatz wird hier nicht klassisch in einen Trainings- und einen Valdierungsdatensatz aufgeteilt, sondern in n komplementäre Mengen unterteilt. Nun kann jedes Netzwerk auf n-1 dieser Datensätze trainiert werden und auf dem letzten validiert werden. Hierdurch gehen die wertvollen Trainingsdaten nicht an die Validierung verloren. Dabei wird n der Anzahl der zu trainierenden Netzwerke gleichgesetzt.
Für die Tiefen Netzwerke N und deren Voraussage für Klasse j P_{ij} gegeben der Eingabe X gilt:

$$ensemble(X)_j = \Sigma_{i=1}^{N} \alpha_i P_{ij}.$$

Wobei α hier die Gewichtung durchführt, um den Durchschnitt zu berechnen.

4.3.4. Ergebnisse

Auf den Validierungsdatensätzen erreichten die Netzwerke eine Top-1 Genauigkeit von 62.1%, wenn es nur darum ging, Pflanzen richtig zu klassifizieren, ohne nicht-bekannte Bildinhalte abzulehnen.
Gemessen anhand der Metrik der PlantCLEF challenge 2016, der mAP-closed, erreichte das ensemble von drei Netzwerken einen Score von 71.0% . Damit wurden Šulc et al. (2016) Dritter des Wettbewerbs. Das Siegerteam von Bluefield erreichte einen offiziellen mAP score von 74.2%, jedoch nutzen sie alle im Testdatensatz vorhandenen Bilder einer Pflanze (*ObservationID*), um eine gemeinsame Vorhersage der Klasse dieser Pflanze zu machen.

4.4. Netzwerkuntersuchung

„Eine der Erkenntnisse des Konnektionismus ist, dass Tiere intelligent werden, wenn viele ihrer Neuronen zusammenarbeiten. Ein einzelnes Neuron oder eine kleine Ansammlung von Neuronen aber nicht besonders nützlich ist “ (Goodfellow et al., 2016, S. 20). Die Grundidee Neuronaler Netzwerke ist an die Funktionsweise des Gehirns angelehnt. Jedes Neuron hat Eingaben und Ausgaben und gewichtet diese. Und auch die Stärke der Gewichtungen wird angepasst. Diese Gemeinsamkeiten mit dem Gehirn machen Neuronale Netzwerke für weitere Untersuchungen interessant. Ich möchte herausfinden, ob weitere Parallelen in der Funktionsweise zu finden sind.

4.4.1. Generalisierung einzelner Layer

Neuronale Netze sind hierarchisch aufgebaut und bestehen in meinem Ansatz aus vielen hintereinander geschalteten Faltungen. Jede dieser Faltungsschichten entspricht einer Anzahl von rezeptiven Feldern. In frühen Schichten sind diese oft einfache Detektoren für Kanten und Ecken. In späteren Schichten können sich auch komplexere rezeptive Felder bilden. Da auch die Einteilung von Pflanzen hierarchisch gestaltet ist, stellt sich die Frage, ob die Netzwerke die gleiche Hierarchie lernen.
Außerdem ist interessant, ab wann Pflanzenorgane gelernt werden. Geschieht dies in frühen Schichten, um die Information der Organe bei der Klassifikation zu nutzen und dann verschiedene Klassifikationswege für verschieden Organe einzuschlagen? Oder werden Organe erst in späten Schichten erkannt?
Des weiteren stellt sich die Frage, ob rezeptive Felder des auf Pflanzen trainierten Netzwerkes andere Eigenschaften haben, als die rezeptiven Felder des auf ImageNet trainierten Netzes. Um dies zu testen muss man die Klassifikationsleistung jeder Schicht messen, der analytische Ansatz hierzu ist eine Regression:

Sei A_{ik} die Matrix aller Bilder/Ausgaben einer Schicht mit *Bild i* und *Pixel k*. S_{ij} sei die Matrix

der Label (Familie, Gattung, Spezies, Pflanzenorgan) mit *Bild i* und *Label j*. Daraus ergibt sich die Matrix der Regressionskoeffizienten B_{kj}. $B_{j_0,k}$ kann als rezeptives Feld beziehungsweise Urbild der Klasse j_0 angesehen werden.
Die lineare Regression setzt sich zusammen aus:

$$S_{ij} = A_{ik} \cdot B_{kj} + \epsilon$$

und wird durch

$$B_{kj} = (A_{ik}^T \cdot A_{ik})^{-1} \cdot A_{ik}^T \cdot S_{ij}$$

gelöst.
$(A_{ik}^T \cdot A_{ik})^{-1} \cdot A_{ik}^T$ ist die *Moore-Penrose Pseudoinverse* der Matrix A und wird benötigt, da die Anzahl der Bilder nicht mit der Anzahl der Pixel übereinstimmen muss.
Das Fehlermaß zwischen der *kleinsten Quadrate Lösung* und den Labeln wird durch

$$||S_{ij} - A_{ik} \cdot B_{kj}||$$

berechnet.
Da dieser Ansatz die (Pseudo-) Inversen sehr großer Matrizen berechnen muss, ist er für Pixelwerte wie sie in diesem Beispiel vorkommen ungeeignet. Beispielsweise ergibt sich für ein 224 x 224 x 3 Pixel großes Bild, welches auf 1000 Klassen trainiert wurde, eine Matrix der Dimensionen 150528 x 1000.
Daher wird auf der Ausgabe jedes zu untersuchenden Layers eine lineare Schicht von Neuronen trainiert, die zur vorhergehenden Schicht voll verknüpft ist. Hierdurch werden die Regressionskoeffizienten über den Gradientenabstieg approximiert, anstatt sie direkt zu berechnen.
Diese Methode ähnelt dem Ansatz von (Mahmood, Bennamoun, An & Sohel, 2016). Sie untersuchten die Generalisierungsfähigkeit der extrahierten *rezeptiven Felder* der Blöcke zwei bis vier. Hierzu nutzten sie ein auf dem ImageNet vortrainiertes ResNet-50 und testeten die Ausgaben der Schichten auf neuen Datensätzen. Anders als Mahmood et al. (2016) habe ich jedoch auf den Ausgaben keine Reduktion der Dimension durchgeführt und auch keine SVM trainiert. Auch Zeiler und Fergus (2013) untersuchten die Schichten ihres Netzwerkes mit SVMs oder voll verbundenen Softmax-Klassifikatoren.
Für das Training der Netzwerke wurde das preprocessing der Trainingsmethode aus Unterabschnitt 4.3.1 verwendet. Die Lernrate wurde auf 0.001, weight decay auf 0.0002 und der Momentum-Term auf 0.9 gesetzt. Die batch-size konnte auf 100 erhöht werden.
Analysiert wurde ein *nicht finegetunetes* Netzwerk, sowie zwei auf Pflanzen trainierte Netze. Bei der Untersuchung habe ich mich auf die Ausgaben der Blöcke eins bis vier (siehe Abbildung 4.5) dieser Netzwerke konzentriert. Für diese Schichten wurden dann 10000 Schritte auf den Trainingsdaten (75470 Bilder) durchgeführt. Somit wurde jedes Bild im Schnitt 13.2 mal betrachtet. Die neue Schicht hat entsprechend der Anzahl der Klassenmitglieder 1000, 516, 124 oder 7 Neurone (Spezies, Genus, Familie, Organ).

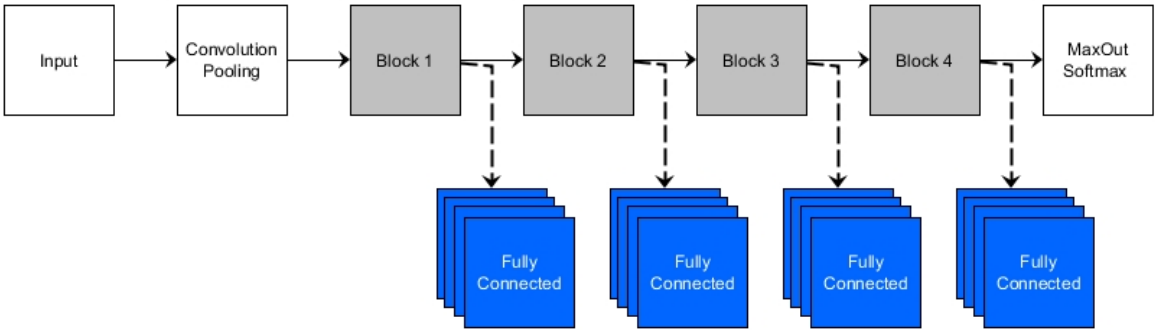

Abbildung 4.5.: Untersuchung der gelernten Merkmalsdetektoren anhand von je vier trainierten, voll verbundenen Neuronenschichten (blau) nach den Blöcken eins bis vier (Familie, Gattung, Spezies, Organ).

4.4.2. Visualisierung und Verständnis der Layer

Die Ausgaben und Aktivierungen einzelner Filter und Layer für verschiedene Bilder zu betrachten ist eine weitere Möglichkeit die Netzwerke zu untersuchen. So nutzen beispielsweise Zeiler und Fergus (2013) eine *deconvolution*, also eine Entfaltung. Dabei werden diejenigen Bilder gesucht, welche einen Filter am stärksten aktivieren, anschließend werden die Pixel hervorgehoben, welche zum Feuern des Neurons beitrugen.
An jede zu untersuchende Schicht wird hierfür ein *Entfaltungs-Netzwerk* (engl. *deconvolutional network*) angelegt. Alle Aktivierungen der Schicht, außer die des zu untersuchenden Filters, werden auf Null gesetzt. Dann wird diese Aktivierung als Eingabe des Entfaltungs-Netzes verwendet. Dieses führt die vom Netzwerk durchgeführten Operationen in umgekehrter Reihenfolge durch. Hierzu ist es nötig, Faltungen, Pooling-Operationen und Nicht-Linearitäten (*rectification*, aufgrund der ReLU [6]) rückgängig zu machen. Dies kann meist nur approximativ geschehen. Maximum-Pooling beispielsweise, ist nicht invertierbar. Merkt man sich allerdings die Positionen der maximalen Aktivitäten, ist es approximiert invertierbar. Faltungen werden invertiert, indem jeder Filter horizontal und vertikal geflippt wird und dann auf die Eingabe angewendet wird. Da das Residuale Netzwerk Nicht-Linearitäten verwendet, muss die *feature map*, also die Eingabe des geflippten Filters, zuvor auch durch die ReLU aktiviert werden.
Yosinski et al. (2015) verbesserten diese Visualisierung. Chu, Yang und Tadinada (2017) führten seine Visualisierungen auf dem ResNet-152 durch, da in letzterem Paper jedoch keine genaueren Angaben zu den genutzten Methoden gegeben sind, konnte ich deren Ansatz nicht nutzen.
Mahendran und Vedaldi (2014) betrachteten die Aktivierung eines Filters wenn ein Bild präsentiert wurde und versuchten dann ein Bild zu rekonstruieren, welches ähnliche Aktivierungsmuster erzeugt.

Ähnlich wie bei Zeiler und Fergus (2013) suche ich als erstes die *neun Bilder, welche einen*

[6] Rectified linear unit: Half-wave rectification

Filter maximal aktivieren. Als Metrik für die Aktivierung wird die Summe der Aktivierung über die komplette feature map verwendet. Nun werden aus jedem Block zufällig drei rezeptive Felder (*Channel*) ausgewählt und abgebildet. Weiterhin werden die Aktivierungen der Filter für diese maximal aktivierenden Bilder zurück auf die Pixelebene projiziert, um zu erkennen, was das Netzwerk „sieht“ und auf was der Filter maximal reagiert. Hierfür nutze ich die oben beschriebene Entfaltung [7].
Des Weiteren wird für jeden Channel untersucht, ob die neun Bilder, welche ihn maximal aktivieren, der gleichen Spezies, Familie, Gattung oder dem selben Organ angehören. So können rezeptive Felder gefunden werden, welche beispielsweise auf Blätter oder Blüten reagieren. Beziehungsweise ob Pflanzen bereits während der Faltung unterschiedlich behandelt werden. Für die erste Faltung werden die Gewichte direkt visualisiert [8].

[7] `https://github.com/InFoCusp/tf_cnnvis`, 11.08.2017
[8] `https://gist.github.com/kukuruza/03731dc494603ceab0c5`, 11.08.2017

5. Ergebnisse

In diesem Kapitel werden zuerst die Ergebnisse meiner Replikation präsentiert. Hierfür werden die Top-1 und Top-5 Genauigkeiten aller meiner trainierten Netzwerke auf den Validierungs- und Trainingsdatensätzen ausgewertet. Außerdem werden Ensembles von Netzwerken, sowie einzelne Netzwerke anhand des mAP-Scores auf dem Testdatensatz verglichen (Abschnitt 5.1 und Abschnitt 5.2). Daraufhin werden in den weiteren Abschnitten die Ergebnisse der Netzwerkuntersuchungen aufgezeigt.

5.1. Validierungsgenauigkeiten

Die von den Netzwerken erreichten Klassifikationsleistung hängen von den gewählten Hyperparametern und Bildvorbearbeitungen während des Trainings ab. Die Netzwerkarchitektur ist für alle Netzwerke gleich. In Tabelle 5.1 sind alle Ergebnisse zu sehen.
Besonders gute Ergebnisse erzielt hier Netzwerk 3, welches auf seinem Validierungsdatensatz von 37735 Bildern eine Genauigkeit von 64.48% erreicht. Dies ist besser als die Ergebnisse des CMP Teams. Da sich die Validierungsdatensätze jedoch unterscheiden, können die Ergebnisse nur bedingt verglichen werden.

Genauigkeit Validierung				
Netzwerk	**Training Top-1**	**Training Top-5**	**Validierung Top-1**	**Validierung Top-5**
Originalpaper ohne MaxOut			62.1%	
Netzwerk 1	72.7%	90.0%	57.44%	78.19%
Netzwerk 2	48.33%	73.40%	36.3%	59.9%
Netzwerk 3	91.40%	98.50%	**64.48%**	82.73%
Netzwerk 4	99.96%	100.00%	47.88%	67.36%
Netzwerk 5	99.96%	100.00%	47.73%	67.16%
Netzwerk 6	94.8%	99.5%	50.90%	70.60%
Netzwerk 7	94.4%	99.4%	52.46%	72.26%
Netzwerk 8 [1]	91.23%	98.66%	54.20%	74.33%

Tabelle 5.1.: Top-1 und Top-5 Genauigkeiten der Netzwerke auf den Trainings- und Validierungsdatensätzen.

5.2. Replikation PlantClef-Challenge

Im Gegensatz zur vorherigen Metrik entsprechen die Ergebnisse hier direkt den Ergebnissen aus dem offiziellen PlantCLEF Wettbewerb, da der selbe Datensatz und das gleiche Auswertungsskript verwendet wurden. Die Resultate können in Tabelle 5.2 betrachtet werden. Zur Übersicht wurden die drei Einreichungen der CMP Gruppe mit in die Tabelle aufgenommen (*CMP Netzwerk **).
Hier erzielt Netzwerk 1, sowie das Ensemble von Netzwerk 6, 7 und 8 die besten Resultate. Der *mAP-closed-world*-Score übersteigt den Wert der besten Einreichung von Šulc et al. (2016). Die Ergebnisse des *mAP-invasive*-Wertes sind nur für Netzwerk 1 besser. Und die *mAP-open-world*-Werte sind für alle getesteten Voraussagen schlechter.

mAP-Scores

Netzwerk	**mAP Closed World**	**mAP Invasive**	**mAP Open World**
CMP Netzwerk 1	0.71	0.653	0.79
CMP Netzwerk 2	0.644	0.564	0.729
CMP Netzwerk 3	0.639	0.59	0.723
Ensemble (Netzwerk 4, 5)	0.659	0.512	0.584
Ensemble (Netzwerk 6, 7, 8)	**0.749**	0.576	0.675
Ensemble (Netzwerk 1, 3)	**0.724**	0.556	0.637
Netzwerk 1	**0.994**	**0.739**	0.697
Netzwerk 3	**0.729**	0.556	0.642

Tabelle 5.2.: Erreichte mAP-Werte der Netzwerke/Ensembles, berechnet mit dem offiziellen Auswertungstool auf dem offiziellen Testdatensatz.

5.3. Generalisierung einzelner Layer

Insgesamt werden aus drei Netzwerken je vier Proben entnommen. Je eine Probe nach jedem Block des Residualen Netzwerkes. Zum Vergleich werden Netzwerk 3 und Netzwerk 5, sowie ein auf ImageNet vortrainiertes Netz herangezogen. Letzteres ist also nicht auf die Bestimmung von Pflanzen trainiert.
Verschieden tiefe Schichten dieser Netzwerke erreichen unterschiedlich gute Top-1 und Top-5 Genauigkeiten auf den Trainings- beziehungsweise Validierungsdatensätzen. Die Ergebnisse der Validierung sind in Tabelle 5.3 aufgeführt und graphisch in Abbildung 5.1 dargestellt. Detaillierte Ergebnisse sind in Tabelle A.1, Tabelle A.2 und Tabelle A.3 im Anhang zu finden. Für alle Klassen, mit Ausnahme der Pflanzenorgane, werden die Ergebnisse der tieferen Schichten immer besser. Dabei werden stets ähnliche Leistungen auf dem Trainings- und Testdatensatz erreicht. Die Werte übersteigen dabei die Chance. Dies gilt auch für die untersten Layer. Für

die Familie beträgt dieser Wert $\frac{1}{124} \approx 0.008$, für die Gattung $\frac{1}{516} \approx 0.0019$, für die Spezies $\frac{1}{1000} = 0.001$ und für die Organe $\frac{1}{7} \approx 0.14$.

Netzwerkvergleich: Analyse der Blöcke 1 - 4 des ResNet-50							
		Netzwerk 5		ImageNet		Netzwerk 3	
Klasse	**Schicht**	**Top 1**	**Top 5**	**Top 1**	**Top 5**	**Top 1**	**Top 5**
Familie	Block 1	0.07	0.24	0.07	0.24	0.08	0.26
Familie	Block 2	0.11	0.33	0.09	0.3	0.1	0.32
Familie	Block 3	0.1	0.34	0.12	0.34	0.11	0.34
Familie	Block 4	0.27	0.58	0.27	0.58	0.24	0.51
Genus	Block 1	0.02	0.05	0.02	0.05	0.02	0.06
Genus	Block 2	0.03	0.11	0.04	0.13	0.04	0.14
Genus	Block 3	0.08	0.2	0.08	0.21	0.09	0.21
Genus	Block 4	0.48	0.72	0.51	0.74	0.38	0.62
Spezies	Block 1	0.003	0.03	0.01	0.04	0.01	0.03
Spezies	Block 2	0.05	0.12	0.04	0.1	0.04	0.1
Spezies	Block 3	0.11	0.22	0.09	0.19	0.08	0.18
Spezies	Block 4	0.53	0.75	0.62	0.81	0.42	0.62
Organ	Block 1	0.33	0.87	0.32	0.88	0.32	0.86
Organ	Block 2	0.32	0.86	0.32	0.88	0.33	0.89
Organ	Block 3	0.33	0.88	0.32	0.89	0.32	0.87
Organ	Block 4	0.31	0.87	0.33	0.88	0.32	0.87

Tabelle 5.3.: Top-1 und Top-5 Genauigkeiten der einzelnen Schichten der Netzwerke auf den Validierungsdatensätzen nach je 10000 Trainingsiterationen.

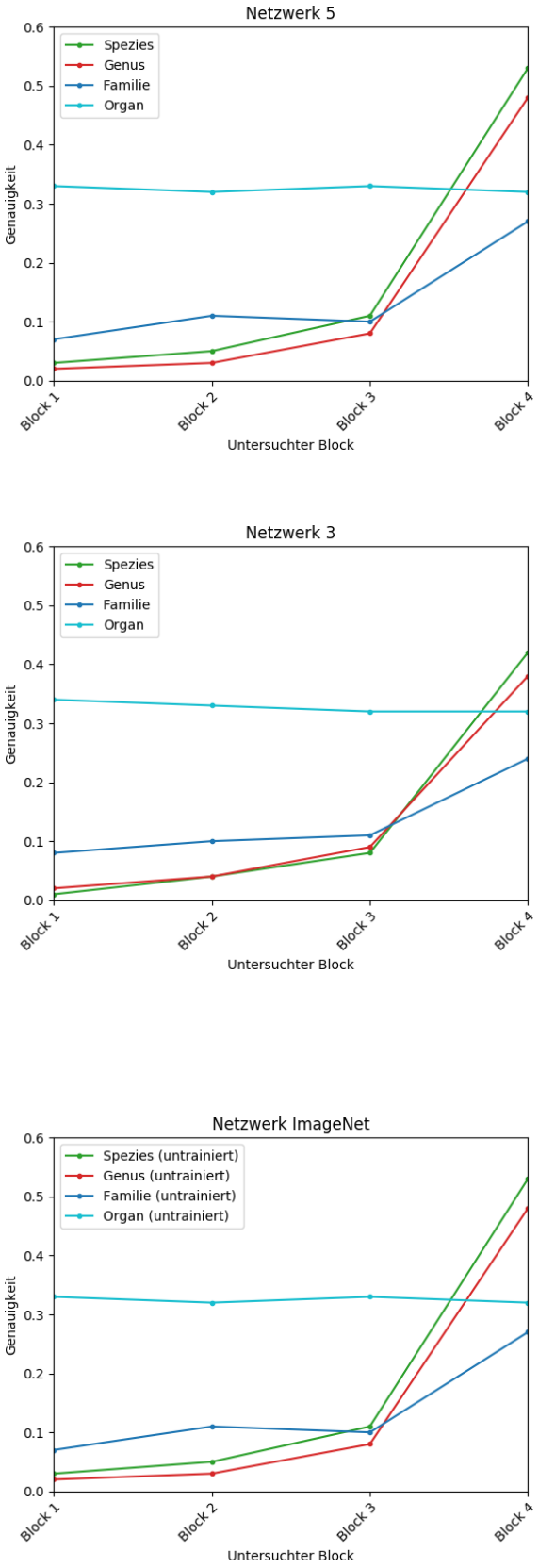

Abbildung 5.1.: Die Top 1 Genauigkeiten der Ausgaben auf dem Validierungsdatensatz. Die Ergebnisse der Ausgaben verschiedener Blöcke sind (von oben nach unten) für Netzwerk 5, Netzwerk 3 und das ImageNet-Netzwerk eingezeichnet.

5.4. Visualisierung der Layer

In diesem Abschnitt werden sowohl die Visualisierungen einzelner Faltungs-Schichten, als auch die Bilder präsentiert, welche einzelne rezeptive Felder maximal aktivieren.

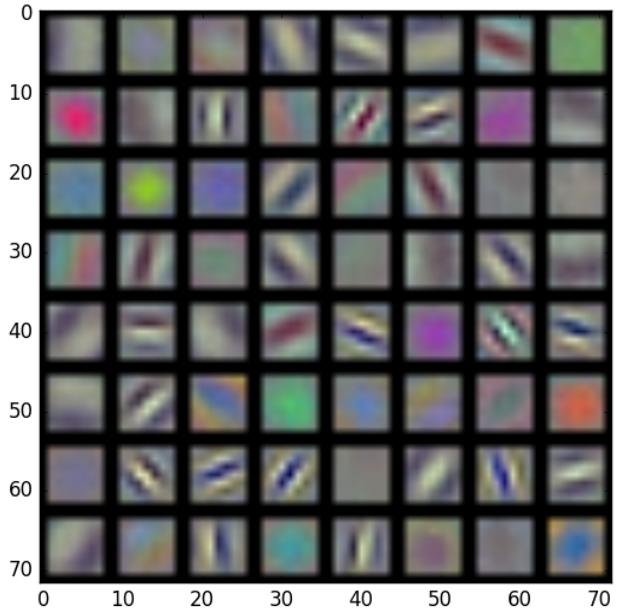

Abbildung 5.2.: Visualisierung des ersten Convolutional Layers.

In Abbildung 5.2 ist die Visualisierung der ersten Faltungsschicht abgebildet. Hier ist es noch möglich die 64 rezeptiven Felder direkt zu visualisieren, da diese Filter direkt auf dem Eingabebild arbeiten. Sie bilden typische Gabor-Filter. Beispielsweise ist in Filter 11 (Zeile 2, Spalte 3) ein Kosinus-Gabor-Filter zu sehen. In Filter 12 (Zeile 2, Spalte 4) hat sich ein Sinus-Gabor-Filter und mit Filter 9 (Zeile 2, Spalte 1) ein isotropischer Gabor-Filter gebildet. Diese Typen von rezeptiven Feldern werden auch im visuellen Sehsystem des Menschen gefunden (Mallot, 2013, S. 49). Es sind also einfache Kanten- und Farbdetektoren entstanden.

Wie in Unterabschnitt 4.4.2 erklärt, können spätere Schichte nur schwer visualisiert werden. Deshalb wurden für sie die *neun* Bilder gesucht, welche einen Filter maximal aktivieren. In Tabelle 5.4 ist die Anzahl der rezeptiven Filter abgetragen, deren neun Bilder *alle* auf eine bestimmte Klasse maximal reagierten.
In Abbildung 5.3, Abbildung 5.4, Abbildung 5.5 und Abbildung 5.6 sind die Ergebnisse dieser Analyse für je drei zufällig gewählte rezeptive Felder aus den Blöcken 1 bis 4 abgebildet. Außerdem wird zu diesen Aktivierungen die Deconvolution nach Zeiler und Fergus (2013), sowie die Aktivierung der feature maps für das entsprechende Bild abgebildet.

Aktivitäten der Channel	
Klasse	**Anzahl Channel**
Familie	91
Genus	61
Spezies	58
Blume	2881
Blatt	0
Blattscan	721
Stamm	30
Frucht	10
Ast	0

Tabelle 5.4.: Anzahl der Filter, deren Top-9 Aktivierungen einem Exemplar der Klasse entsprechen.

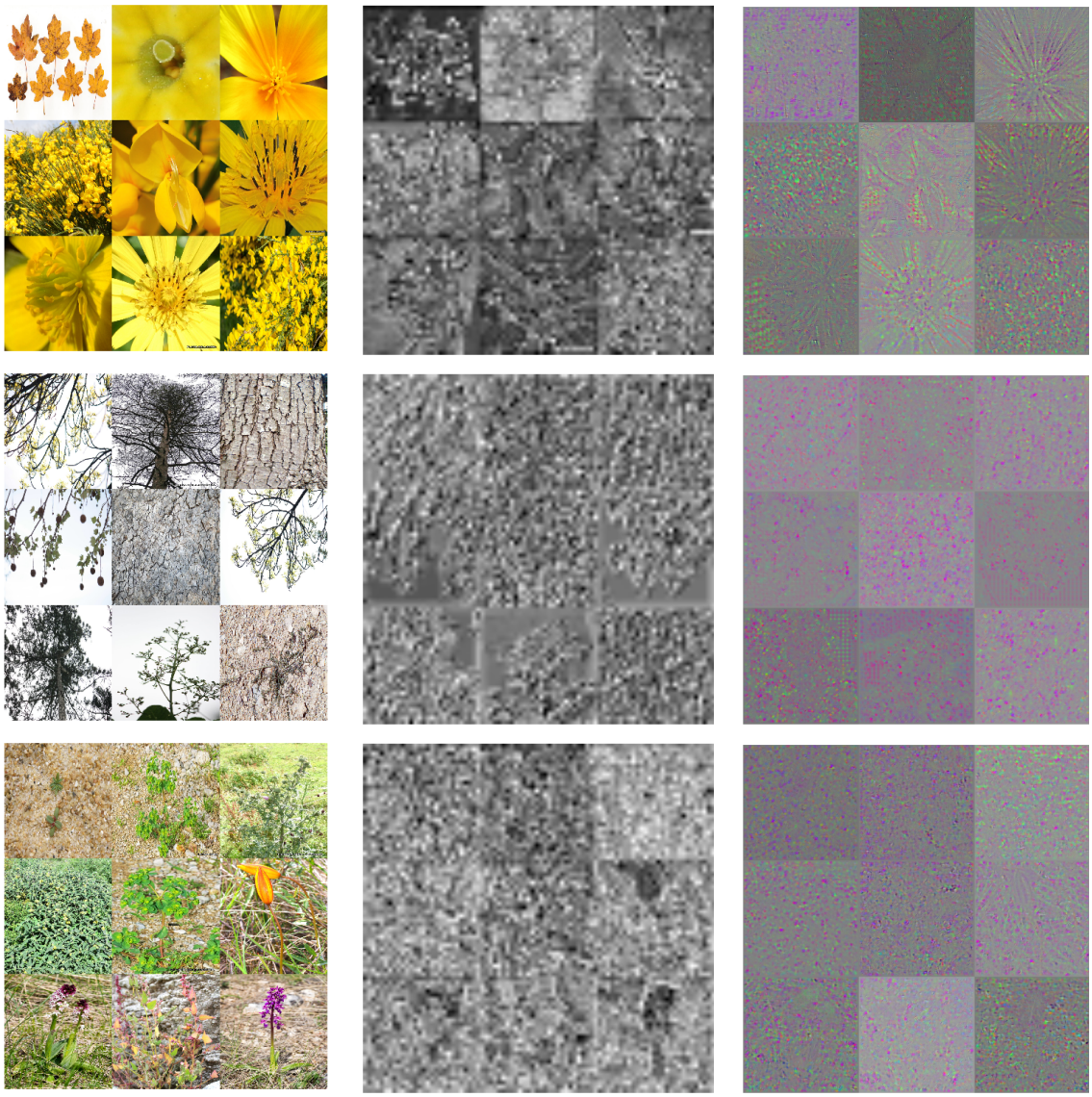

Abbildung 5.3.: *Links:* Die neun Bilder, welche die rezeptiven Felder *14, 25 und 71* (von oben nach unten) in **Block 1** Bottleneck 3 maximal aktivieren. *Mitte:* Die feature map bei Eingabe des Bildes. *Rechts:* Die Aktivitäten des Filters zurückpropagiert auf die Eingabeebene (Deconvolution nach **Zeiler et al.**)

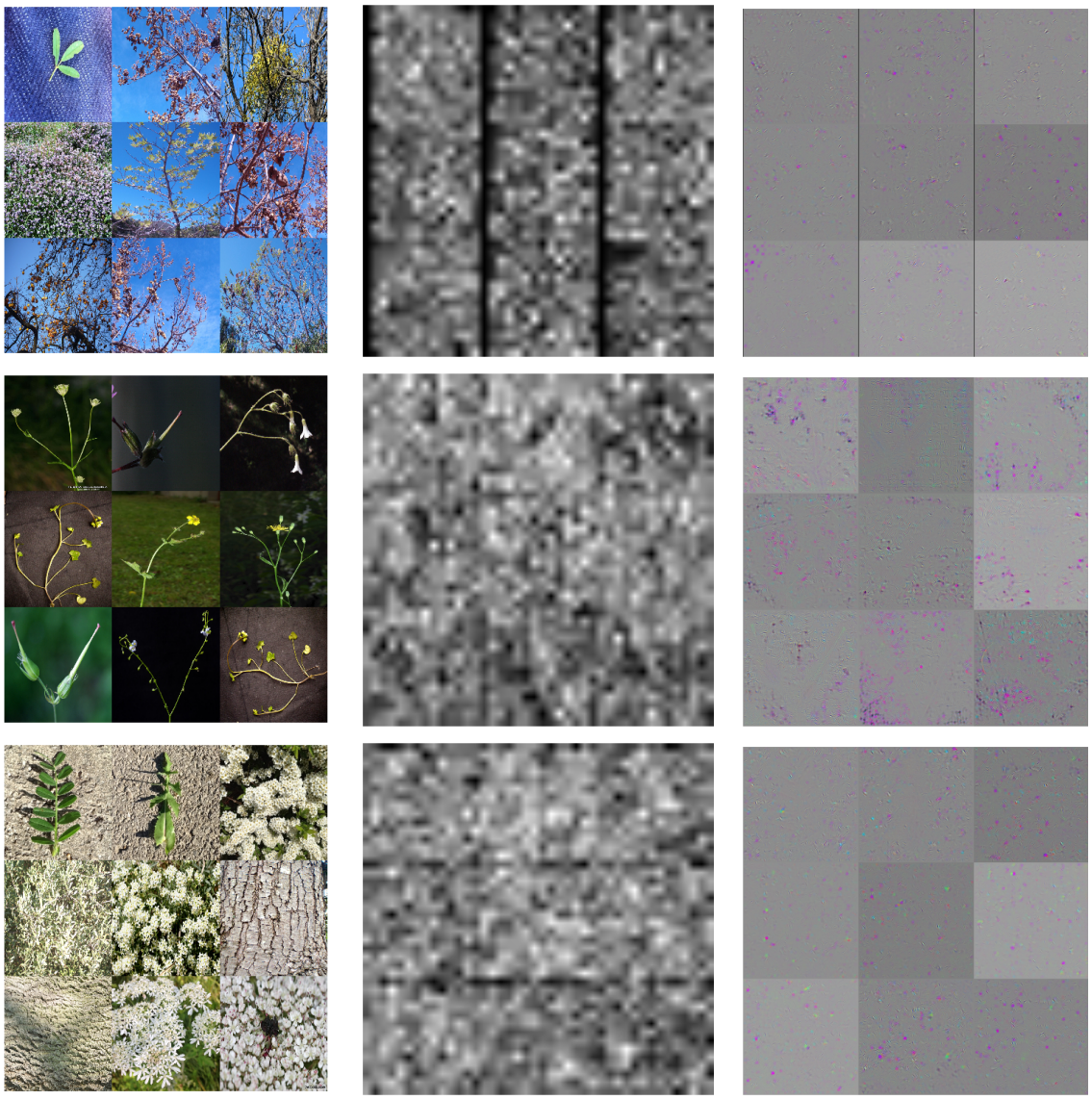

Abbildung 5.4.: *Links:* Die neun Bilder, welche die rezeptiven Felder **0, 47 und 79** (von oben nach unten) in **Block 2** Bottleneck 4 maximal aktivieren. *Mitte:* Die feature map bei Eingabe des Bildes. *Rechts:* Die Aktivitäten des Filters zurückpropagiert auf die Eingabeebene (Deconvolution nach **Zeiler et al.**)

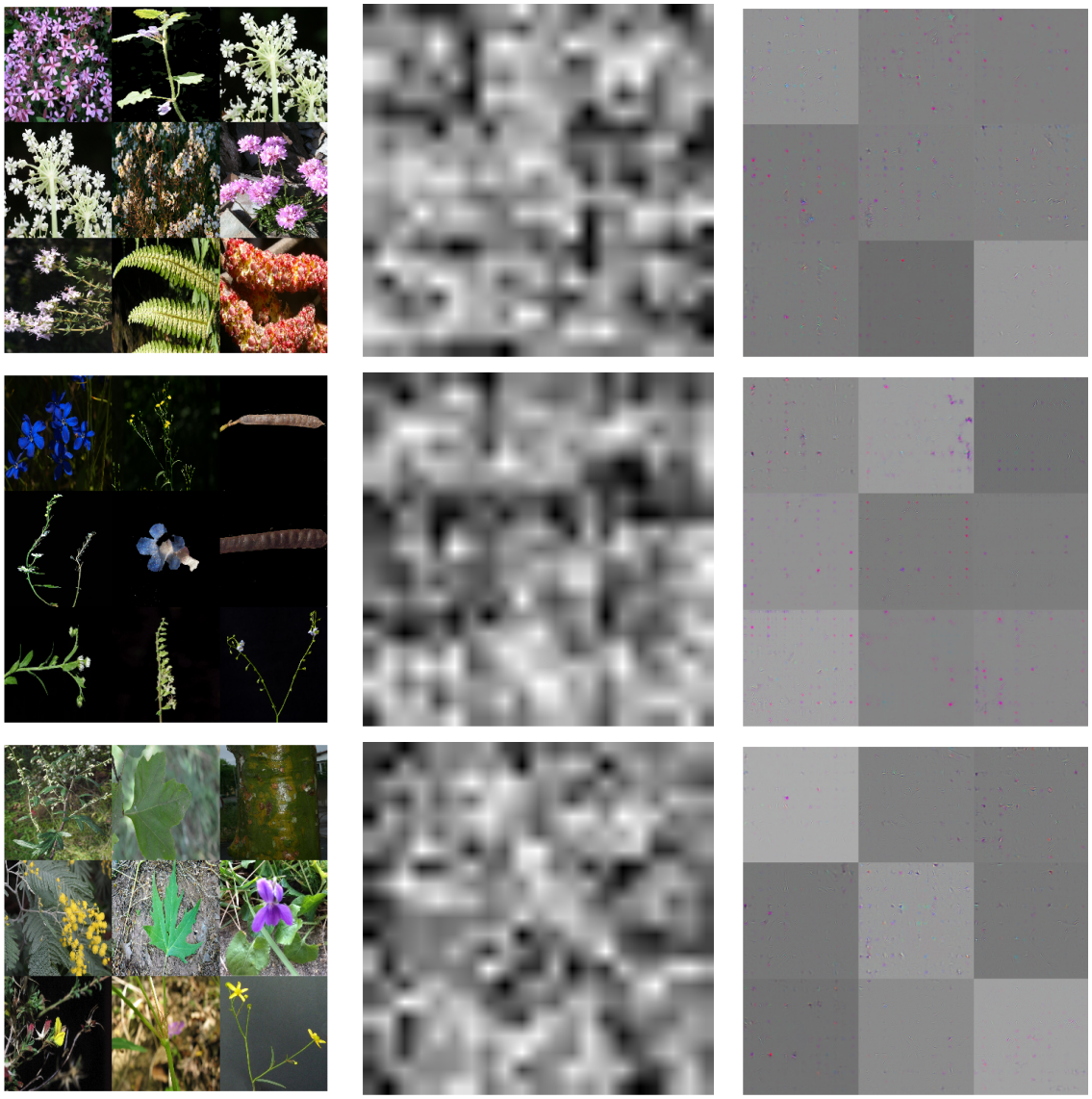

Abbildung 5.5.: *Links:* Die neun Bilder, welche die rezeptiven Felder **14, 21 und 597** (von oben nach unten) in **Block 3** Bottleneck 6 maximal aktivieren. *Mitte:* Die feature map bei Eingabe des Bildes. *Rechts:* Die Aktivitäten des Filters zurückpropagiert auf die Eingabeebene (Deconvolution nach **Zeiler et al.**)

Abbildung 5.6.: *Links:* Die neun Bilder, welche die rezeptiven Felder **9, 12 und 35** (von oben nach unten) in **Block 4** Bottleneck 3 maximal aktivieren.

6. Diskussion

Die Ergebnisse aus Abschnitt 5.1 und Abschnitt 5.2 zeigen, dass die Replikation nur teilweise gelungen ist. Netzwerk 1 erzielt zwar 99.4%, beziehungsweise 73.9% Präzision auf den „brute-force“ Klassifizierungs-Aufgaben, schafft es jedoch nicht, unbekannte Klassen abzulehnen. Ähnliches gilt für das Ensemble von Netzwerk 6, 7 und 8, sowie Netzwerk 3. Im Vergleich zur CMP-Gruppe wurden bessere Ergebnisse für die **mAP closed-world**-Metrik, jedoch schlechtere Ergebnisse für die **mAP open-world**-Metrik und die **mAP invasive**-Metrik erzielt. Letztere sind darauf zurückzuführen, dass die Ensembles wohl zu ähnliche Voraussagen abgegeben haben.
Überraschend sind die vergleichsweise guten Validierungsergebnisse der *Netzwerke 1 und 2*, deren Farbwerte zufällig variiert wurden. Das Netzwerk scheint sich weniger auf Farben, als auf die Form der Pflanze beschränkt zu haben. Außergewöhnlich sind auch die schlechten Genauigkeiten der *Netzwerke 6, 7 und 8*. Die Normalisierung der Farbwerte, sowie die zufällige Rotation der Bilder scheint nicht für genügend Variation gesorgt zu haben. Das Netzwerk lernte den Trainingsdatensatz zu gut und generalisiert zu schlecht. Dieses „overfitting“ ist auch daran zu erkennen, dass Netzwerk 8 nach 102500 Schritten bessere Ergebnisse erzielt, als die Netzwerke 6 und 7.
Die Lernrate für *Netzwerk 2* war deutlich zu gering. Die niedrigen Top-1 und Top-5 Genauigkeit auf den Trainingsdaten würden weitere Trainingsschritte erlauben.
Insgesamt bleibt festzuhalten, dass Netzwerk 3 zu weiteren Untersuchungen gut geeignet ist, da es gute Ergebnisse für den Validierungsdatensatz lieferte. Es ist also fähig, Pflanzen zu klassifizieren.

Betrachtet man die Generalisierungsleistung aus Abschnitt 5.3, erkennt man, dass die Klassifikationsleistung erst in Block 4 ansteigt. Diese Resultate stimmen mit den Ergebnissen von Mahmood et al. (2016) und den Erkenntnissen von Garcia-Gasulla et al. (2017) überein. Letztere behaupten, dass rezeptive Felder in frühen und mittleren Schichten deskriptive und Filter in späteren Schichten diskriminierende Aufgaben haben. Deskriptoren helfen bei der Unterscheidung nur wenig. Die Leistungen aller drei Netzwerke für die ersten drei Blöcke stimmen nahezu überein, obwohl die trainierten Netzwerke unterschiedlich gute Erkennungsraten haben, beziehungsweise ein Netzwerk überhaupt nicht auf die Klassifikation von Pflanzen trainiert wurde. Dies spricht dafür, dass die rezeptiven Felder früher Schichten schon genügend ausgebildet sind, um auch auf unbekannten Aufgaben genutzt werden zu können (Mahmood et al., 2016). Außerdem scheint die Pflanzenklassifikation keine besonderen Anforderungen an frühe rezeptive Felder zu stellen, obwohl die Unterscheidung hier auf kleinsten Unterschieden beruht. Dies ist auch in Abbildung 5.2 zu sehen. Die dort abgebildeten rezeptiven Felder der

ersten Schicht des Netzwerks zeigen die typischen Gabor-Filter Eigenschaften. Hier sind keine sichtlichen Anpassungen an die Aufgabe der Pflanzenklassifikation geschehen. All diese Ergebnisse sprechen dafür, dass Neuronale Netzwerke in frühen Operationen Bildinformationen entnehmen, die für allen Arten von Bildern genutzt werden können.

Das Training des Softmax Klassifikators auf Netzwerk 3, also dem Netzwerk mit der höchsten Genauigkeit, liefert die schlechtesten Ergebnisse für die Generalisierung. Diese Resultate erkläre ich mir mit der Annahme, dass die rezeptiven Felder sich zu „Deskriptoren“ zurückentwickelt haben. Dies könnte damit zusammenhängen, dass die Unterschiede zwischen den verschiedenen Pflanzenklassen zu gering waren, um gute Unterscheidungen treffen zu können. Betrachtet man nun die Bilder, welche ein rezeptives Feld maximal aktivieren, erkennt man deutlich, dass gewisse Bildeigenschaften gelernt wurden. Block 1 (Abbildung 5.3) scheint auf einfache Bildelemente zu reagieren. Filter *14* beispielsweise auf die Farbe gelb und Filter *71*, sowie Filter *25* auf viele feine Kanten. Auch in der *Entfaltung* (Deconvolution) von Filter 14 erkennt man, dass das Netzwerk stark auf die Kanten mit gelber Färbung zu reagieren scheint. Die rezeptiven Felder am Ende der Blöcke 2 und 3 reagieren weiter auf einfache Bildeigenschaften.
Die Filter aus Block 3 scheinen auf den Hintergrund zu achten. In Block 4 reagieren die rezeptiven Felder bereits auf komplexere und konjugierte Muster. Channel *9* reagiert auf Blüten in lila Farben, Channel *12* auf Blätter-Scans mit weißem Hintergrund und Filter *35* scheint auf feine Strukturen vor hellem Hintergrund zu reagieren, dies sind hauptsächlich Nadeln von Bäumen. Die Entfaltungen der Netzwerke sind leider nur für die frühen Faltungen aussagekräftig und nur für die ersten drei Blöcke vorhanden. Danach sind die Visualisierungen nicht mehr interpretierbar.

Die hohe Erkennungsrate der Blüten, die in Tabelle 5.4 zu sehen ist, kann auf die Farbigkeit dieser zurückgeführt werden. Somit reagieren auch schon frühe, einfache rezeptive Felder stark auf sie. Mit diesen Ergebnissen wird auch der Erkennungswert von Organen während der Schichten-Generalisierung (≈ 0.32) verständlich. Blüten werden erkannt, somit sind bereits $\frac{1}{7}$ des Wertes aufgeklärt. Für die restlichen sechs Organe rät das Netzwerk mit einer Chance von $\frac{1}{6}$. Addiert man diese Werte, so ergibt sich ein Wert von ≈ 0.309. Diese Erkenntnis ist besonders überraschend, da in Block 4 scheinbar Filter entstanden sind, welche maximal auf Organe reagieren. Auch die 721 erkannten Blatt-Scans sprechen dafür, dass sich hierfür bestimmte Filter gebildet haben. Diese scheinen jedoch nicht in ausreichender Zahl vorhanden zu sein, um die Ausgaben der Netzwerke zu beeinflussen.

Die Ergebnisse sprechen dafür, dass das tiefe Residuale Netzwerk von Šulc et al. (2016) die Taxonomie von Pflanzen *nicht* lernt. Die Filter lernen zu abstrahieren und entwickeln auch sinnvolle rezeptive Felder, jedoch scheinen die visuellen Daten nicht genügend Informationen zu liefern, um eine Unterteilung in Familien oder Gattungen durchzuführen. Das ist daran zu erkennen, dass die Erkennungsrate der einzelnen Blöcke mit zunehmender Abstraktionsstufe abnimmt. Dies liegt vermutlich daran, dass Pflanzen innerhalb einer Familie zu große Unterschiede in ihrem Erscheinungsbild haben. Verstärkt wird dieser Effekt dadurch, dass für alle

Pflanzenarten alle sieben Organ-Teile präsentiert werden. Ich vermute, dass sich die Leistung bei gleichbleibender Anzahl von Trainingsdaten drastisch verbessert, falls sich auf ein Organ konzentriert wird. Dieser Ansatz wird bereits erfolgreich von Applikationen wie Leafsnap genutzt (Kumar et al., 2012). Dieses Erkennungstool nutzt nur Fotos von Blättern mit weißem Hintergrund und erreicht damit sehr gute Erfolge.

Um die Ergebnisse der Netzwerke zu verbessern, sollten die Bildvorbearbeitungsmethoden (*preprocessing*) genauer analysiert werden. Dieser Prozess entscheidet mit darüber, ob das Netzwerk *overfitted* oder nicht. Bisher scheint es, als würden die zufällige Variation der Farben die Leistung eher verbessern, als verschlechtern. Da die Organe nicht gelernt wurden, stellt sich die Frage, ob explizit definierte, getrennte Klassifikationswege im Netzwerk die Leistung verbessern würden. Wie oben bereits erklärt, wäre diese Netzwerkarchitektur jedoch nur bei genügend Trainingsdaten pro Klasse sinnvoll.
Um die Netzwerke weiter zu untersuchen könnte nach Zeiler und Fergus (2013) eine Heatmap für die Bereiche erstellt werden, welche das Netzwerk betrachtet. Außerdem kann die Generalisierung der verschiedenen Blöcke weiter ausgebaut werden. Erstens zeigen die niedrigen Genauigkeiten auf den Trainingsdatensätzen, dass weitere Trainingsepochen möglich sind. Diese wurden aus zeitlichen Gründen nicht mehr weiter durchgeführt. Jedoch lassen sich erst dann endgültige Aussagen über ihre Generalisierungsleistung tätigen, wenn die Netzwerke voll trainiert sind. Zweitens sollten die Untersuchungen nach allen Faltungen durchgeführt werden. Insbesondere in Block 4 steigen die Genauigkeiten exponentiell an. Um den Rechenaufwand zu minimieren, kann auf das Verfahren von Mahmood et al. (2016) zurückgegriffen werden. Hier werden zuerst die Dimensionen anhand einer *Principle Component Analysis* verringert und anschließend eine *Support-Vektormaschine* auf den verkleinerten Eingaben trainiert.
Eine weitere sinnvolle Untersuchung ist die Methode von Garcia-Gasulla et al. (2017). Die Autoren verglichen die *Intra*-Klassen-Aktivierung mit der *Inter*-Klassen-Aktivierung für jedes rezeptive Feld. Sie berechneten hierzu den Quotienten D_{KS} der beiden Werte. Ist der $D_{KS} \approx 0$, bedeutet dies, dass die Aktivierung eines rezeptiven Feldes für alle Bilder der Klasse c_i gleich der Aktivierung für alle Bilder der Klassen $c_{j \neq i}$ ist. Somit hilft dieses rezeptive Feld nicht bei der Unterscheidung von Klassen, kann jedoch für die Beschreibung sinnvoll sein. Zum Beispiel wird ein Kantendetektor in fast jedem Bild aktiv sein. Hohe D_{KS} zeigen die Bildeigenschaften auf, die speziell in einer Klasse vorkommen. Auf Pflanzen bezogen könnte dies zum Beispiel eine besondere Farbe sein. Extrem niedrige D_{KS} sind jedoch auch hilfreich. Da sie die Abwesenheit gewisser Merkmale in Klasse c_i zeigen, die in den anderen Klassen $c_{j \neq i}$ vorhanden sind. Dies hilft bei der Unterscheidung von Klassen. Mittels dieser Metrik könnte genauer untersucht werden, ab wann die Filter des Residualen Netzes diskriminierend arbeiten.
Die Untersuchung der „Urbilder“ einer Klasse ist eine weitere Analysemethode. Für jede Klasse gibt das Netzwerk eine durch die Softmax-Funktion $\sigma(x_j) = \frac{e^x_j}{\sum_{k=1}^{K} e^{x_k}}$ gewichtete Ausgabe zurück, sodass sich die Summe aller Ausgaben pro Bild zu 1 aufsummiert. Bestenfalls sollte das Netzwerk somit *1* für die richtige Klasse j und 0 für alle anderen Klassen $k \neq j$ ausgeben. In der Praxis geschieht dies jedoch nie. Verändert man ein Bild aus Rauschen anhand des Gradienten immer weiter, entsteht ein Urbild einer Klasse. Das entstandene Urbild gibt Aufschluss darüber, ob das Netzwerk die richtigen Eigenschaften des Bildes gelernt hat. Der DeepDream

Generator des *Inceptionism Teams* [1] agiert auf den Gradienten und optimiert den Input. Hierfür generiert er Bilder, welche die Summe der Aktivitäten eines rezeptiven Feldes maximieren. Als Ausgangspunkt für den Gradientenabstieg wurde ein verrauschtes, graues Bild verwendet. Da die daraus resultierenden Bilder hauptsächlich hohe Frequenzen enthalten, wird eine *Laplace-Pyramiden-Gradienten-Normalisierung* (engl. *Laplacian Pyramid Gradient Normalization*) durchgeführt, sodass die niedrigen und mittleren Frequenzen auch auf den Bildern hervortreten können.

Abschließend bleibt festzuhalten, dass ich zu viel Zeit auf die Replikation der Ergebnisse und die verschiedenen Bildvorverarbeitungs-Methoden verwendet habe, weshalb für die Untersuchung des Netzwerkes nicht genügend Zeit zur Verfügung stand. Dies lag unter anderem an den spärlichen Angaben in den Papern von Šulc et al. (2016), sowie der langen Einarbeitungsphase in Tensorflow. Ein weiterer limitierender Faktor war das Training, da dieses für ein tiefes Neuronalen Netz sehr viel Zeit in Anspruch nimmt. Durch Nutzung des BW-Clusters, auf dem viele Ansätze parallel ausgewertet werden können, wurde dieses Problem in den letzten Tagen meiner Bachelorarbeit jedoch verringert.
Nichtsdestotrotz zeigen meine Ergebnisse, dass Tiefe Neuronale Netzwerke wichtige Charakteristiken von Pflanzen lernen und extrahieren können. Zu einer besseren Klassifizierung von Pflanzen anhand von Fotografien müssen die allgemein genutzten Klassifikationswege jedoch explizit definiert werden. Diese beinhalten die getrennte Betrachtung von verschiedenen Organen und die Kombination ihrer Ergebnisse, wie sie zum Beispiel in Pflanzenbestimmungswerken wie „Flora von Deutschland und seinen angrenzenden Gebieten“ (Schmeil & Fitschen, 2011) durchgeführt werden.

[1] `https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/tutorials/deepdream/deepdream.ipynb`, 11.08.2017

Literaturverzeichnis

Chu, B., Yang, D. & Tadinada, R. (2017). Visualizing residual networks. *arXiv:1701.02362v1*.

Garcia-Gasulla, D., Parés, F., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., ... Suzumura, T. (2017). On the behavior of convolutional nets for feature extraction. *arXiv:1703.01127v2*.

Ghazi, M., Yanikoglu, B. & Aptoula, E. (2016). Open-set plant identi cation using an ensemble of deep convolutional neural networks. *Working notes of CLEF 2016 conference*.

Goëau, H., Bonnet, P. & Joly, A. (2016). Plant identifcation in an open-world (lifeclef 2016). *CLEF working notes 2016*.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT Press. (`http://www.deeplearningbook.org`)

Goodfellow, I., Warde-Falrey, D., Mirza, M., Courville, A. & Bengio, Y. (2013). Maxout networks. *arXiv:1302.4389v4*.

Hang, S., Tatsuma, A. & Aono, M. (2016). Bluefield (kde tut) at lifeclef 2016 plant identification task. *Working notes of CLEF 2016 conference*.

He, K., Zhang, X., Ren, S. & Sun, J. (2015, December). Deep residual learning for image recognition. *arXiv:1512.03385v1*.

Hinton, G., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2012, July). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Joly, A., Goëau, H., Glotin, H., Spampinato, C., Bonnet, P., Vellinga, W., ... Müller, H. (2016). Lifeclef 2016: Multimedia life species identifcation challenges. *Proceedings of CLEF 2016*.

Krizhevsky, A., Sutskever, I. & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *NIPS*.

Kumar, N., Belhumeur, P., Biswas, A., Jacobs, D., Kress, W., Lopez, I. & Soares, J. (2012). Leafsnap: A computer vision system for automatic plant species identification. *Proc. ECCV*.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998, November). Gradient-based learning applied to document recognition. *Proc. of the IEEE*.

Lin, M., Chen, Q. & Yan, S. (2014). Network in network. *arXiv:1312.4400v3*.

Mahendran, A. & Vedaldi, A. (2014). Understanding deep image representations by inverting them. *arXiv:1412.0035v1*.

Mahmood, A., Bennamoun, M., An, S. & Sohel, F. (2016). Resfeats: Residual network based features for image classification. *arXiv:1611.06656v1*.

Mallot, H. (2013). *Computational neuroscience a first course* (Bd. 2; N. Kasabov, Hrsg.). Springer.

Riesenhuber, M. & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2* (11), 1019 - 1025.

Schmeil, O. & Fitschen, J. (2011). *Die Flora von Deutschland und der angrenzenden Länder. Ein Buch zum Bestimmen aller wild wachsenden und häufig kultivierten Gefäßpflanzen* (S. Seybold, Hrsg.). Quelle & Meyer.

Serre, T., Kouh, M., Cadieu, C., Knoblich, U., Kreiman, G. & Poggio, T. (2005). A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex. *AI Memo 2005-036/CBCL Memo 259, Massachusetts Inst. of Technology, Cambridge*.

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M. & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 411 – 426.

Simonyan, K. & Zisserman, A. (2015, April). Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556v6*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2014, September). Going deeper with convolutions. *arXiv:1409.4842v1*.

Thorpe, S., Fize, D. & Marlot, C. (1996, June). Speed of processing in the human visual system. *Letters To Nature*, *381*, 520 - 522.

Šulc, M., Mishkin, D. & Matas, J. (2016). Very deep residual networks with maxout for plant identifcation in the wild. *Working notes of CLEF 2016 conference*.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. & Lipson, H. (2015). Understanding neural networks through deep visualization. *arXiv:1506.06579v1*.

Zeiler, M. & Fergus, R. (2013, November). Visualizing and understanding convolutional networks. *arXiv:1311.2901v3*.

A. Anhang

A.1. Weitere Ergebnisse: Generalisierung der Layer

Netzwerk 5: Analyse der Blöcke des ResNet-50

Klasse	Schicht	Set 1 Top1	Set 1 Top5	Set 2 Top1	Set 2 Top5	Set 3 Top1	Set 3 Top5
Familie	Block 1	0.06	0.22	0.06	0.23	0.07	0.24
Familie	Block 2	0.1	0.32	0.1	0.32	0.11	0.33
Familie	Block 3	0.1	0.33	0.1	0.35	0.1	0.34
Familie	Block 4	0.25	0..53	0.27	0.57	0.27	0.58
Genus	Block 1	0.01	0.05	0.01	0.06	0.02	0.05
Genus	Block 2	0.07	0.19	0.06	0.16	0.03	0.11
Genus	Block 3	0.13	0.3	0.09	0.22	0.08	0.2
Genus	Block 4	0.4	0.61	0.56	0.78	0.48	0.72
Spezies	Block 1	0.01	0.03	0.01	0.04	0.003	0.03
Spezies	Block 2	0.08	0.18	0.04	0.11	0.05	0.12
Spezies	Block 3	0.1	0.21	0.13	0.24	0.11	0.22
Spezies	Block 4	0.45	0.66	0.64	0.84	0.53	0.75
Organ	Block 1	0.34	0.86	0.33	0.87	0.33	0.87
Organ	Block 2	0.32	0.86	0.33	0.88	0.32	0.86
Organ	Block 3	0.35	0.88	0.33	0.89	0.33	0.88
Organ	Block 4	0.34	0.88	0.32	0.88	0.31	0.87

Tabelle A.1.: Top-1 und Top-5 Genauigkeiten von Netzwerk 5 für einzelne Schichten auf den Trainings- (Set 1 und Set 2) und Validierungsdatensätzen (Set 3) nach je 10000 Iterationen.

ImageNet: Analyse der Blöcke des ResNet-50							
Klasse	**Schicht**	**Set 1 Top1**	**Set 1 Top5**	**Set 2 Top1**	**Set 2 Top5**	**Set 3 Top1**	**Set 3 Top5**
Familie	Block 1	0.07	0.25	0.07	0.24	0.07	0.24
Familie	Block 2	0.1	0.29	0.09	0.29	0.09	0.3
Familie	Block 3	0.12	0.37	0.13	0.37	0.12	0.34
Familie	Block 4	0.25	0..53	0.27	0.57	0.27	0.58
Genus	Block 1	0.01	0.05	0.01	0.06	0.02	0.05
Genus	Block 2	0.05	0.14	0.04	0.14	0.04	0.13
Genus	Block 3	0.11	0.27	0.11	0.25	0.08	0.21
Genus	Block 4	0.41	0.62	0.54	0.77	0.51	0.74
Spezies	Block 1	0.02	0.05	0.01	0.04	0.01	0.04
Spezies	Block 2	0.05	0.13	0.05	0.12	0.04	0.1
Spezies	Block 3	0.13	0.27	0.13	0.26	0.09	0.19
Spezies	Block 4	0.44	0.0.66	0.57	0.78	0.62	0.81
Organ	Block 1	0.34	0.87	0.33	0.88	0.32	0.88
Organ	Block 2	0.32	0.88	0.32	0.88	0.32	0.88
Organ	Block 3	0.32	0.89	0.33	0.89	0.32	0.89
Organ	Block 4	0.33	0.88	0.31	0.87	0.33	0.88

Tabelle A.2.: Top-1 und Top-5 Genauigkeiten ImageNet-Netzes für einzelne Schichten auf den Trainings- (Set 1 und Set 2) und Validierungsdatensätzen (Set 3) nach je 10000 Iterationen.

Netzwerk 3: Analyse der Blöcke des ResNet-50

Klasse	Schicht	Set 1 Top1	Set 1 Top5	Set 2 Top1	Set 2 Top5	Set 3 Top1	Set 3 Top5
Familie	Block 1	0.08	0.23	0.08	0.25	0.08	0.26
Familie	Block 2	0.1	0.33	0.1	0.32	0.1	0.32
Familie	Block 3	0.1	0.34	0.12	0.33	0.11	0.34
Familie	Block 4	0.22	0.49	0.24	0.53	0.24	0.51
Genus	Block 1	0.01	0.23	0.02	0.06	0.02	0.06
Genus	Block 2	0.04	0.33	0.05	0.14	0.04	0.14
Genus	Block 3	0.09	0.34	0.09	0.22	0.09	0.21
Genus	Block 4	0.35	0.49	0.42	0.64	0.38	0.62
Spezies	Block 1	0.01	0.05	0.01	0.03	0.01	0.03
Spezies	Block 2	0.04	0.12	0.05	0.12	0.04	0.1
Spezies	Block 3	0.09	0.21	0.11	0.21	0.08	0.18
Spezies	Block 4	0.37	0.57	0.46	0.67	0.42	0.62
Organ	Block 1	0.32	0.87	0.31	0.87	0.34	0.86
Organ	Block 2	0.35	0.88	0.32	0.87	0.33	0.89
Organ	Block 3	0.33	0.88	0.33	0.88	0.32	0.87
Organ	Block 4	0.33	0.87	0.32	0.87	0.32	0.87

Tabelle A.3.: Top-1 und Top-5 Genauigkeiten von Netzwerk 3 für einzelne Schichten auf den Trainings- (Set 1 und Set 2) und Validierungsdatensätzen (Set 3) nach je 10000 Iterationen.

A.2. mAP Metrik

In einer kurzen Untersuchung habe ich die Einreichungen der CMP Gruppe, sowie die Voraussagen meines Ensembles der Netzwerke 6, 7 und 8 mit einem Schwellenwert angepasst. Dieses Verfahren wurde bereits von der Sabanci Gruppe durchgeführt (Ghazi et al., 2016). Dabei wurden alle Voraussagen, die eine Softmax-Ausgabe unter 0.33 hatten auf 0 gesetzt. Hierdurch erreichte mein Netzwerk 85.7%, 65.5%, sowie 76.1% Präzision für die mAP-Metriken. Die beste Einreichung der CMP-Gruppe erreichte 98.3%, 94.7% sowie 93.5% Präzision.
Da ich hierbei den Testdatensatz, sowie die genaue Metrik bereits zur Verfügung hatte, um einen guten Schwellenwert zu finden, ist diese Untersuchung hinsichtlich des PlantCLEF-Wettbewerbs nicht besonders aussagekräftig. Jedoch zeigt es meiner Meinung nach auf, dass die mAP-Metrik für Menschen nicht intuitiv zu verstehen ist. 85.7% Präzision auf der *closed-world* Aufgabe implizieren sehr gute Erkennungsraten für bekannte Pflanzenarten. Vergleicht man diesen Wert jedoch mit den Validierungsgenauigkeiten der einzelnen Netzwerke (54.2%), so erkennt man die Diskrepanzen. Als Vergleich zwischen den einzelnen Systemen ist die Metrik jedoch trotzdem sinnvoll.