Früchte und Gemüse Klassifizierung mit CNN

CAS Machine Intelligence FS 2022

Vanessa Bruder, Marc Hauswirth, Nicole Lehnherr

School of Engineering

1. Einleitung

Dieses Projekt wurde im Rahmen des Moduls «Deep Learning» im CAS Machine Intelligence erarbeitet.
Übergreifendes Ziel war es, sich vertieft mit der Materie vom Deep Learning auseinanderzusetzen und erste Erfahrungen im Umgang mit Modellen und dem Code in Python zu sammeln.
In diesem Fall wurde mittels CNN (Convolutional Neural Networks) ein Modell darauf trainiert, welches Bilder von Früchten und Gemüse aus rund 131 unterschiedlichen Klassen richtig zuordnen kann.
Mit dem durch die Projektgruppe trainierten Modell wurde eine Accuracy auf den Test-Daten von rund 97% erreicht, was ein guter Wert ist.

2. Ziel und Methode

Ziel war es, mittels Deep Learning ein Modell zu entwickeln, welches dazu in der Lage ist, Bilder von Früchten und Gemüse in 131 verschiedene Klassen einzuteilen.
Um die Performance zu überprüfen, wurden zwei Baselines erstellt:

1. Baseline: zufällige Zuordnung zu allen Klassen
   Im ersten Schritt wurde ein einfaches Modell herbeigezogen, welches von der Annahme ausgeht, dass jedes Bild mit derselben Wahrscheinlichkeit in jede Klasse eingeteilt werden kann. Jede Klasse wird also mit der Wahrscheinlichkeit von 1/131 für jedes Bild ausgewählt.

2. Baseline: VGG16 – ImageNet
   Als zweite Baseline wurde das bereits trainierte Modell VGG16 (ImageNet) verwendet. Dieses Modell wurde von Somonyan, und Zisserman, (2015) entwickelt. Trainiert wurde es auf rund 1.3 Mio. Bildern mit 1000 Klassen. Das VGG16, welches in der Keras Bibliothek verfügbar ist, wurde auf die Bilder des hier verwendeten Datasets angewendet, um einen Vergleich zu haben, wie gut die Performance von einem bereits trainierten Modell ist.

Für das hier vorliegende Projekt wurde ein Convolutional Neural Network trainiert, was als «State of the Art»-Netzwerk gilt, wenn es sich um Bilddaten handelt. Im Convolutional Layer werden nebeneinander liegende Pixel zusammengefasst und erhalten geteilte Gewichte.

3. Herausforderungen

Gerechnet wurde mittels Jupyter Notebook auf Google Colab. Eine zentrale Herausforderung war es somit die Rechenleistung und somit die Geschwindigkeit zu optimieren. Es wurden verschiedene Wege getestet, die Bilddateien einzulesen, was zu massiv unterschiedlichen Performances in punkto Geschwindigkeit geführt hat. Schlussendlich war es klar die schnellste Variante, einen Zip-Ordner mit den Bildern direkt in die Umgebung von Google Colab zu importieren und so auf die Files zuzugreifen.

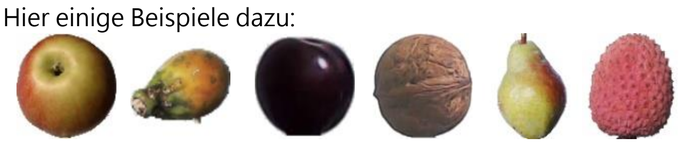
4. Dataset

Als Dataset wurde ein Archiv von Bildern verwendet welches über Kaggle verfügbar ist:

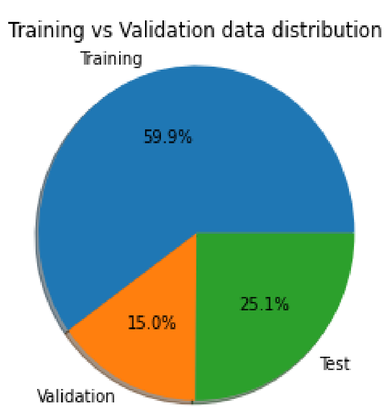
https://www.kaggle.com/datasets/moltean/fruits

Der Datensatz besteht aus 90 380 Bildern, welche in 131 Klassen von Früchten und Gemüse unterteilt sind. Klassen sind zum Beispiel Haselnuss, Grapefruit Pink, Apfel Braeburn und Apfel Granny Smith.

Die Bildgrösse beträgt 100x100 Pixel. Die Bilder wurden vor einem weissen Hintergrund aufgenommen und auch noch vom Hintergrund extrahiert, damit nur die Früchte darauf ersichtlich sind. Es handelt sich um eine 360 Grad-Aufnahme einer Frucht. Hier einige Beispiele dazu:



Der Datensatz wurde unterteilt in Training, Validation und Test. Knapp 60% der Bilder wurden für das Training verwendet, was 54 190 Bildern entspricht. Für die Validation wurden 13 502 Bilder verwendet und für den Test 22 688.



5. Baseline Modelle

Um zuerst ein Verständnis der Daten, respektive deren Wahrscheinlichkeit richtig zugeordnet zu werden, zu erhalten, haben wir zu Beginn zwei Baselines erstellt.

Einfache Wahrscheinlichkeit und NLL

Zu Beginn hatten wir noch wenig Wissen zu unserem Trainings-Dataset und sind daher von einer Wahrscheinlichkeit von 0.0076 (1 zu 131) ausgegangen. Das heisst, jedes Bild hat dieselbe Wahrscheinlichkeit richtig zugordnet zu werden. Der negative log likelihood (NLL) liegt dabei bei 4.875. Diesen haben wir gemäss der Formel berechnet:

nr_of_classes=131
-np.log(1/nr_of_classes)

Wir haben damit ungefähr dasselbe Resultat erhalten, wie mit der Funktion model.evaluate für das noch nicht trainierte CNN, welches bei 4.882 lag.

VGG16 – ImageNet

In einem zweiten Schritt haben wir dann das vortrainierte Modell VGG16 verwendet, um zu sehen, wie die Wiedererkennung auf einem vortrainierten Modell ist. VGG16 setzt sich aus 13 convolutional layers, 5 max-pooling layers, und 3 fully connected layer zusammen. Deshalb auch die Zahl 16 im Namen, da sich das Netz aus 16 «tunable» Parameter zusammensetzt, wobei 13 convolution sind und 3 fully connected.

Als Input habe wir 5 Früchte verwendet, wobei 3 wiedererkannt wurden und 2 nicht. Somit war die Wiedererkennungsrate schon bei 60%. Detaillierte Angaben zu den einzelnen Werten sind in der Tabelle zu finden.

Bild	Bezeichnung	Resultat von VGG16	Kommentar
	Apple Red 2	('n03134739', 'croquet_ball', 0.4264004), ('n03720891', 'maraca', 0.3356729), ('n12768682', 'buckeye', 0.09184255), ('n12267677', 'acorn', 0.063574836), ('n07742313', 'Granny_Smith', 0.044215385)	Modell predicted Apfel zu 42% als Krocketball und nur 4% als Apfel Granny Smith.
	Pineapple	('n12768682', 'buckeye', 0.35958377), ('n12267677', 'acorn', 0.3506412), ('n13052670', 'hen-of-the-woods', 0.1408705), ('n07753275', 'pineapple', 0.06873885), ('n07760859', 'custard_apple', 0.030442119)	Modell predicted Ananas zu 35% als Rosskastanie und nur zu 6% als Ananas
	Orange	('n07747607', 'orange', 0.8481453), ('n07749582', 'lemon', 0.14380422), ('n07742313', 'Granny_Smith', 0.001831102), ('n07753592', 'banana', 0.0012129848), ('n07684084', 'French_loaf', 0.0010300735)	Modell predicted Orange zu 84% als Orange.
	Strawberry	('n07745940', 'strawberry', 0.98684967), ('n12620546', 'hip', 0.0017366698), ('n03720891', 'maraca', 0.0010359043), ('n03729826', 'matchstick', 0.00089072634), ('n07579787', 'plate', 0.0006446182)	Modell predicted Erdbeere zu 98% als Erdbeere
	Banana	('n07753592', 'banana', 0.85032374), ('n03495258', 'harp', 0.041748635), ('n03532672', 'hook', 0.022975475), ('n03047690', 'clog', 0.009548296), ('n01930112', 'nematode', 0.009439148)	Modell predicted Banane zu 85% als Banane

6. Modell

Unser finales Modell besteht aus folgenden Layers und Parametern:

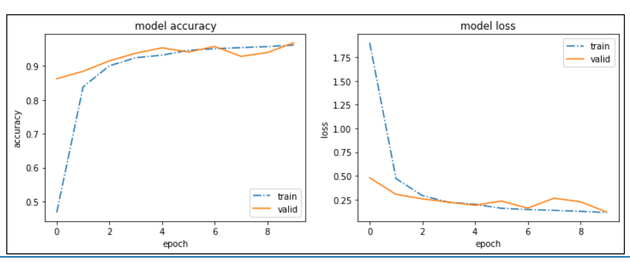
- zwei Convolution Layers
- einem Maxpooling Layer
- zwei fully connected dense Layers zu 500 und 300
- dropout mit 0.3
- flatten
- Softmax
- Kernel size von 10, 10
- Image size von 100, 100
- Pool size von 2,2

Zudem verwendeten wir padding= « same» um sicherzustellen, dass der Filter auf allen Elementen des Inputs ausgeführt wird. Relu als Standard activation für deep neural networks, so dass wir dem Problem mit dem verschwindenden Gradienten entgegenwirken.

```
model  =  Sequential()

model.add(Conv2D(16,(3,3),activation="relu",padding="same",input_shape=(100,100,3)))
model.add(Conv2D(16,(3,3),activation="relu",padding="same"))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(32,(3,3),activation="relu",padding="same"))
model.add(Conv2D(32,(3,3),activation="relu",padding="same"))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dropout(0.3))
model.add(Dense(500))
model.add(Activation('relu'))
model.add(Dropout(0.3))
model.add(Dense(300))
model.add(Activation('relu'))

model.add(Dense(131))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

Accuracy und Loss Training / Validation

Unser Model weist am Schluss kein over/underfitting mehr auf. Die Trainingskurve verläuft ähnlich wie die Validierungskurve.



7. Jupyter Notebook

https://github.com/hauswmar/CAS-Deep-Learning---Projekt/blob/main/DL_Projekt_FruitCNN_V1.0.ipynb

https://github.com/hauswmar/CAS-Deep-Learning---Projekt/blob/main/DL_Project_Baseline_VGG16.ipynb

8. Resultat

Unser finales CNN Modell ist in der Lage die Früchtedaten zu einer hohen Wahrscheinlichkeit richtig zu klassifizieren.
Während der Erarbeitung des Modells wurden viele Möglichkeiten entdeckt, um das Modell zu verbessern.

Parameter die das Resultat beeinflussten

- Anzahl der Convolutional Layers
- Anzahl Neuronen in den Convolutional Layers
- Batch Size
- Kernel Size
- Anzahl Dense Layers
- Anzahl Neuronen in den Dense Layers
- Optimizer «Adam» oder «SGD»
- Dropout hinzugefügt
- Data Augmentation hinzugefügt
- Daten normalisiert

Accuracy und Loss der verschiedenen Modelle

Ergebnis nach 10 Epochen Training der Validierungsdaten

	Acc	Loss
Baseline Einfache Wahrscheinlichkeit	0.007600	4.887500
CNN Einfach	0.936756	0.244679
CNN Komplexer I	0.967336	0.116320
CNN Komplexer II	0.940476	0.199289
CNN Komplexer III	0.921503	0.311193
CNN Komplexer III mit Dropout	0.910937	0.320437
CNN Komplexer I mit Dropout	0.968676	0.119068

Die beste Performance konnte mit dem Modell «**CNN Komplexer I mit Dropout**» erzielt werden. Die untenstehenden Werte basieren auf diesem Modell.

Klassifikations-Report der ersten 15 Klassen

Classification Report	precision	recall	f1-score	support
Apple Braeburn	0.69	0.59	0.64	164
Apple Crimson Snow	1.00	0.97	0.99	148
Apple Golden 1	0.80	1.00	0.89	160
Apple Golden 2	0.94	1.00	0.97	164
Apple Golden 3	1.00	1.00	1.00	161
Apple Granny Smith	1.00	1.00	1.00	164
Apple Pink Lady	1.00	1.00	1.00	152
Apple Red 1	0.98	1.00	0.99	164
Apple Red 2	1.00	0.85	0.92	164
Apple Red 3	0.97	1.00	0.99	144
Apple Red Delicious	1.00	1.00	1.00	166
Apple Red Yellow 1	1.00	0.88	0.94	164
Apple Red Yellow 2	1.00	1.00	1.00	219
Apricot	0.88	1.00	0.93	164
Avocado	1.00	1.00	1.00	143
Avocado ripe	1.00	1.00	1.00	166

Accuracy und Loss der Testdaten

	Test accuracy	Test loss
Accuracy & Loss - Testdata	0.972056	0.116203

Trainingsdauer der Modelle

- Das Training dieses Modelles dauerte 30 Minuten

9. Fazit

Nach intensiven Arbeiten an unserem CNN Modell, konnten wir einen interessanten Einblick in die Deep Learning Welt gewinnen. Die Zeit vergeht extrem schnell, wenn man am deep learnen ist. Schlussendlich haben wir einiges gelernt und auch bemerkt, dass noch viel unklar ist und wir gerne noch intensiver in diese Materien eintauchen möchten.

Was bleibt uns in Erinnerung

- Daten einlesen ist nicht ganz trivial
- Daten auf Colab selber speichern und nicht auf Google Drive
- Daten in die richtige Form zu bringen ist nicht ganz einfach

- Mit Tensorflow geht alles viel einfacher als von Hand
- Lernen von Modellen kann sehr zeitintensiv sein
- Freie GPU im Free Colab ist begrenzt und wird bei zu intensiver Nutzung von Google für einige Zeit gesperrt
- Die Investition von 10 Dollar für Colab Pro hat sich gelohnt
  - Schneller GPUs
  - Mehr RAM
  - Macht mehr Spass, wenn es schneller geht

10. Quellen

Somonyan, K. & Zisserman, A. (2015): Very Deep Convolutional Networks for Large-scale Image Recognition. Conference Paper at ICLR 2015

https://tensorchiefs.github.io/dl_course_2022/

https://www.tensorflow.org/api_docs/python/tf/keras/

https://www.kaggle.com/datasets/moltean/fruits/code