Plankton-Image-Classifier

Bourges Julian, Hahn Sandro, Schmalzl Maximilian

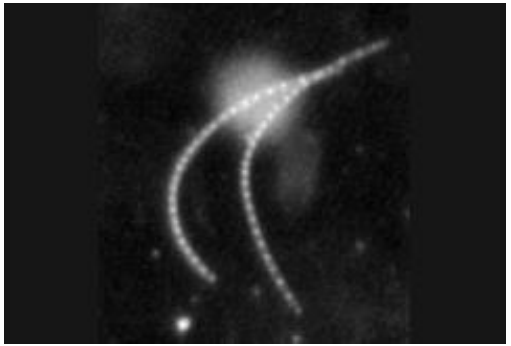
Aufgabenstellung:

- Gegeben sind gelabelte Bilder von Plankton-Spezies
- Problem: Klassen sind deutlich „unbalanced“

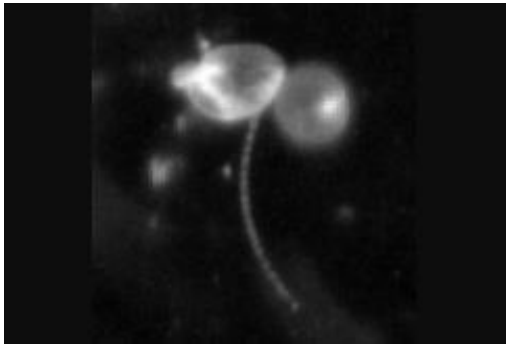
- Aufgabe:
- Verwendung mehrerer Verfahren, möglichst gute Klassifikation, Validierung und Evaluation

Daten

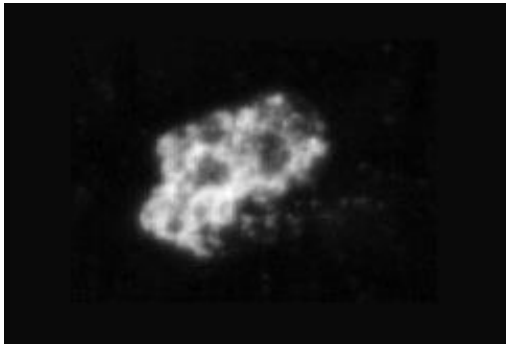
- ~219.000 .png Dateien
- 124.000 train, 95.000 test



Diatom



Noctiluca roi



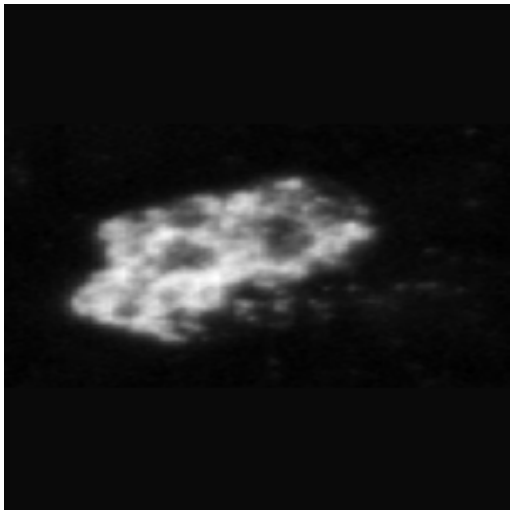
Snow

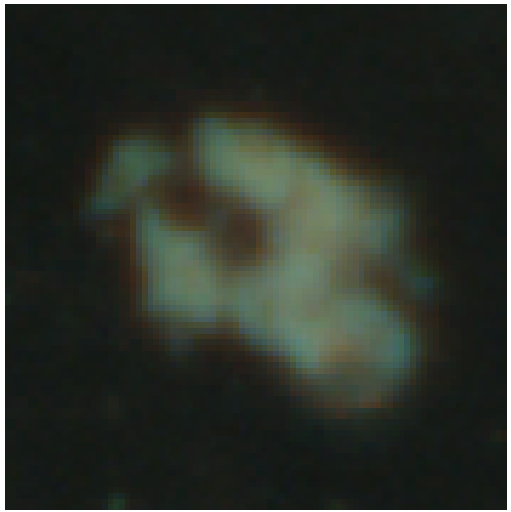
Zoea larvae

Problem:

Test-Bilder haben einen leichten Grünstich

=>Klassifikation schwierig





Datensatz

Problem:

- Plankton-Spezies unterschiedlich stark vertreten

Lösung:

- Eigenen Train-Test Split
- Over- & Undersampling
- Data Augmentation

0	snow	81523	snow	61772
1	pluteus	17558	pluteus	11416
2	diatoms	8335	diatoms	3552
3	copepods	2695	copepods	2997
4	worms	2510	worms	1655
5	blurry	1416	blurry	2428
6	appendicularia_house	999	appendicularia_house	1373
7	noctiluca	995	noctiluca	4064
8	polychaeta	957	polychaeta	401
9	mnemiopsis	882	mnemiopsis	152
10	appendicularia	650	appendicularia	373
11	unknown	608	unknown	1142
12	bipinnaria	565	bipinnaria	98
13	eggs	496	eggs	5
14	medusae	470	medusae	244
15	malacostraca	449	malacostraca	152
16	zoea_larvae	327	zoea_larvae	59
17	rod	315	rod	2870
18	veliger_larvae	297	veliger_larvae	81
19	larvae	275	larvae	23
20	actinotrocha	248	actinotrocha	16
21	pilidium	169	pilidium	75
22	pteropods	701	pteropods	*0*
23	amphipods	288	amphipods	*0*
24	phaeocystis	267	phaeocystis	*0*
25	echinodermata	119	echinodermata	*0*

Output

```
--- test ---
snow=61772
pluteus=11416
noctiluca=4064
diatoms=3552
copepods=2997
rod=2870
blurry=2428
worms=1655
appendicularia_house=1373
unknown=1142
polychaeta=401
appendicularia=373
medusae=244
malacostraca=152
mnemiopsis=152
bipinnaria=98
veliger_larvae=81
pilidium=75
zoea_larvae=59
larvae=23
actinotrocha=16
eggs=5
IntSummaryStatistics{count=22, sum=94948, min=5, average=4315,818182, max
```

```
--- training ---
snow=81523
pluteus=17558
diatoms=8335
copepods=2695
worms=2510
blurry=1416
appendicularia_house=999
noctiluca=995
polychaeta=957
mnemiopsis=882
pteropods=701
appendicularia=650
unknown=608
bipinnaria=565
eggs=496
medusae=470
malacostraca=449
zoea_larvae=327
rod=315
veliger_larvae=297
amphipods=288
larvae=275
phaeocystis=267
actinotrocha=248
pilidium=169
echinodermata=119
IntSummaryStatistics{count=26, sum=124114, min=119, average=4773,615385, max=81
```

Problem

Bilder haben verschiedene Größen

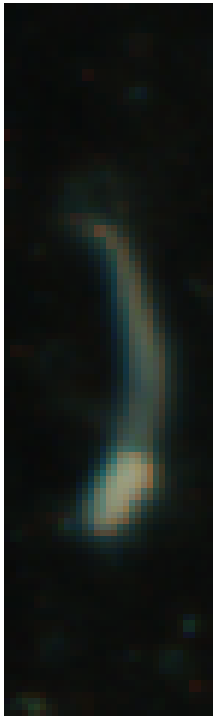
=> Resizing auf Größe 200x200

```
Widths:
0: 21592 (9,86%)
1: 72812 (33,24%)
2: 93711 (42,78%)
3: 15954 (7,28%)
4: 7324 (3,34%)
5: 3609 (1,65%)
6: 1838 (0,84%)
7: 973 (0,44%)
8: 558 (0,25%)
9: 317 (0,14%)
10: 213 (0,10%)
11: 74 (0,03%)
12: 34 (0,02%)
13: 53 (0,02%)

Heights:
0: 19926 (9,10%)
1: 60609 (27,67%)
2: 99519 (45,43%)
3: 19706 (9,00%)
4: 9451 (4,31%)
5: 4675 (2,13%)
6: 2415 (1,10%)
7: 1297 (0,59%)
8: 692 (0,32%)
9: 380 (0,17%)
10: 287 (0,13%)
11: 47 (0,02%)
12: 20 (0,01%)
13: 38 (0,02%)
```

Resizing

```java
public static void resizeImage(int targetWidth, int targetHeight, File input, File output) throws Exception {

    BufferedImage bufferedImage = ImageIO.read(input);

    boolean fitHeight = bufferedImage.getHeight() > bufferedImage.getWidth();
    Scalr.Mode scaleMode = fitHeight ? Scalr.Mode.FIT_TO_HEIGHT : Scalr.Mode.FIT_TO_WIDTH;

    BufferedImage scaledImage = resizeImage(bufferedImage, targetWidth, targetHeight, scaleMode);

    int y_val = (int) ((targetHeight/2.0D) - (scaledImage.getHeight()/2.0D));
    int x_val = (int) ((targetWidth/2.0D) - (scaledImage.getWidth()/2.0D));

    BufferedImage newImage = new BufferedImage(targetWidth, targetHeight, BufferedImage.TYPE_INT_RGB);
    Graphics2D graphics2D = newImage.createGraphics();
    graphics2D.setPaint(new Color( r: 0,  g: 0,  b: 0));
    graphics2D.fillRect( x: 0,  y: 0, newImage.getWidth(), newImage.getHeight());

    if(fitHeight) {
        graphics2D.drawImage(scaledImage, x_val,  y: 0,  observer: null);
    } else {
        graphics2D.drawImage(scaledImage,  x: 0, y_val,  observer: null);
    }

    graphics2D.dispose();
    ImageIO.write(newImage,  formatName: "png" , output);

}
```

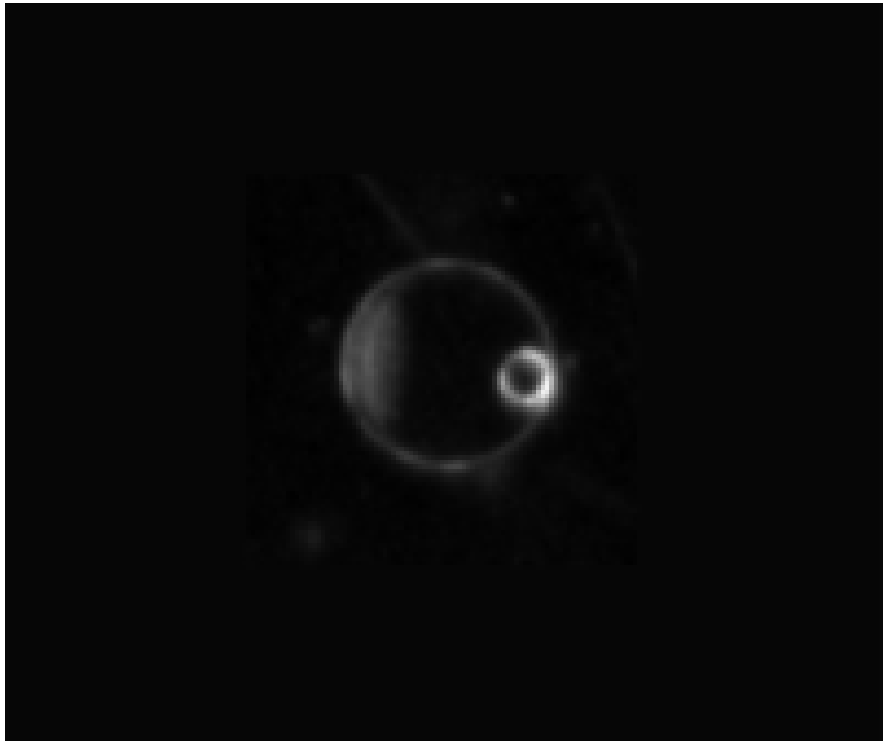
Datenaufbereitung mit Java

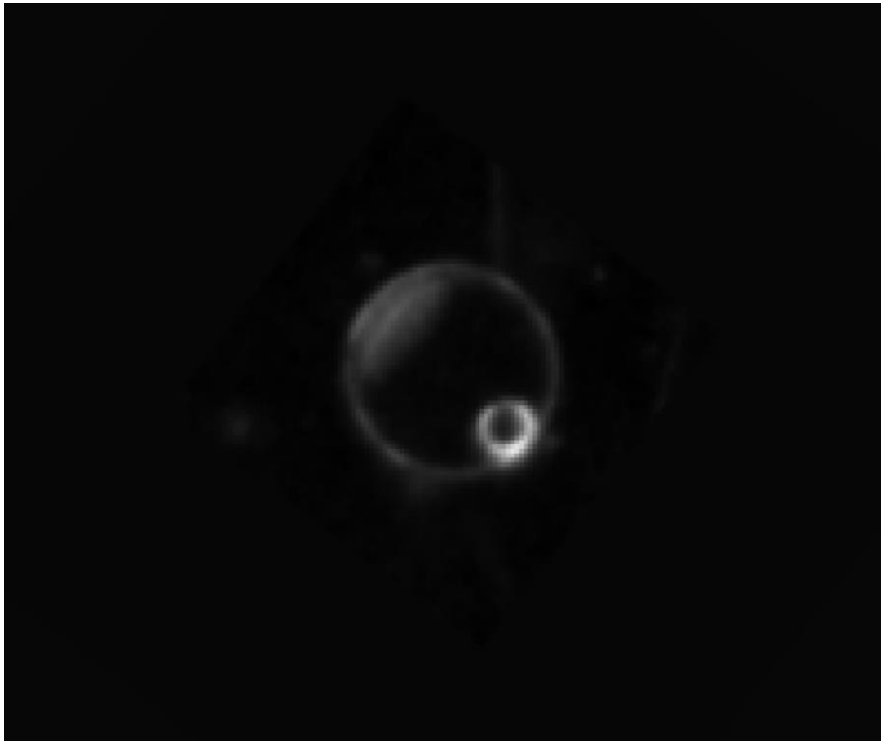


Data Augmentation

```
# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10, # Rotation
        width_shift_range=0.2, # horizontaler Shift
        height_shift_range=0.2, # vertikaler Shift
        zoom_range=0.2, # zoom
        horizontal_flip=True) # flipping
```

Data Augmentation Beispiel





"Fertiger" Datensatz

Training Data			Test Data	
Class ID	Name	Amount	Name	Amount
0	snow	1000	snow	200
1	pluteus	1000	pluteus	200
2	diatoms	1000	diatoms	200
3	copepods	1000	copepods	200
4	worms	1000	worms	200
5	blurry	1000	blurry	200
6	appendicularia_house	1000	appendicularia_house	200
7	noctiluca	1000	noctiluca	200
8	polychaeta	1000	polychaeta	200
9	mnemiopsis	1000	mnemiopsis	200
10	appendicularia	1000	appendicularia	200
11	unknown	1000	unknown	200
12	bipinnaria	1000	bipinnaria	200
13	eggs	1000	eggs	200
14	medusae	1000	medusae	200
15	malacostraca	1000	malacostraca	200
16	zoea_larvae	1000	zoea_larvae	200
17	rod	1000	rod	200
18	veliger_larvae	1000	veliger_larvae	200
19	larvae	1000	larvae	200
20	actinotrocha	1000	actinotrocha	200
21	pilidium	1000	pilidium	200

- Trainingsmenge: 22.000
- Testmenge: 4.400

Decision Trees

Decision Trees

```
[9]
trScore = tree.score(trainX, trainY, sample_weight=None)
testScore = tree.score(testX, testY, sample_weight=None)

print("Train score:", trScore, "; Test Score:", testScore)
```

```
Train score: 0.02272727272727728 ; Test Score: 0.02272727272727728
```

Neural Network

```
root_path = "D:/KI_Plankton/Output200-TrainTest/"

training_files = [f for f in sorted(listdir(root_path)) if "training" in f]
image_list = list()

for file_name in training_files:
    #image_list.append(imread(root_path + file_name).astype('uint8')) #does not work because dtype is float (uses too much ram)
    image_list.append(np.asarray(Image.open(root_path + file_name)))

trainings_arr = np.asarray(image_list)
print(len(trainings_arr))
print(trainings_arr.shape)
print(trainings_arr.dtype)
```

```
22000
(22000, 200, 200, 3)
uint8
```

Save Trainings Data

```
np.save("D:/KI_Plankton/trainings_arr", trainings_arr)
```

Labels einlesen

```
files = [f for f in sorted(listdir(root_path)) if "training" in f]
label_list = list()

for file_name in files:
    if "snow" in file_name:···
    elif "pluteus" in file_name:···
    elif "diatoms" in file_name:···
    elif "copepods" in file_name:···
    elif "worms" in file_name:···
    elif "blurry" in file_name:···
    elif "appendicularia_house" in file_name:···
    elif "noctiluca" in file_name:···
    elif "polychaeta" in file_name:···
    elif "mnemiopsis" in file_name:···
    elif "appendicularia0" in file_name:···
    elif "unknown" in file_name:···
    elif "bipinnaria" in file_name:···
    elif "eggs" in file_name:···
    elif "medusae" in file_name:···
    elif "malacostraca" in file_name:···
    elif "zoea_larvae" in file_name:···
    elif "rod" in file_name:···
    elif "veliger_larvae" in file_name:···
    elif "larvae0" in file_name:···
    elif "actinotrocha" in file_name:···
    elif "pilidium" in file_name:
        label_list.append([21])

training_labels_arr = np.asarray(label_list, dtype='uint8')
print(training_labels_arr)
print(training_labels_arr.shape)
```

```
[[20]
 [20]
 [20]
 ...
 [16]
 [16]
 [16]]
(22000, 1)
```

1-hot-encoding

1-hot-encoding of labels

```
train_labels = np.load("D:/KI_Plankton/trainings_labels_arr.npy")
test_labels = np.load("D:/KI_Plankton/test_labels_arr.npy")

# one hot encode target values
trainY = to_categorical(train_labels)
testY = to_categorical(test_labels)

print(np.shape(trainY))
print(np.shape(testY))
```

```
(22000, 22)
(4400, 22)
```

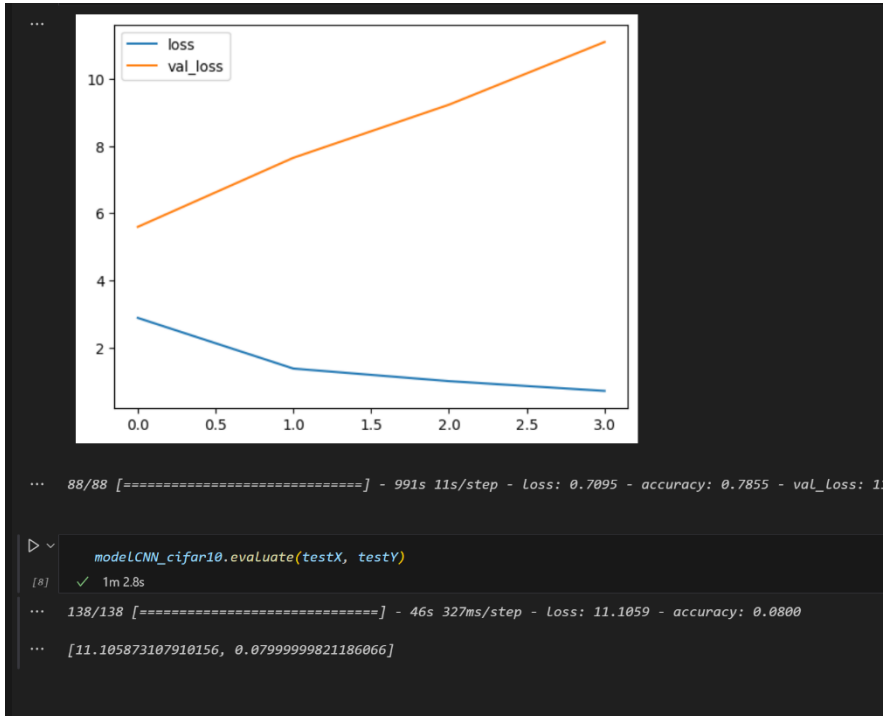
Quelle : https://www.youtube.com/watch?v=BecEHOVmx9o

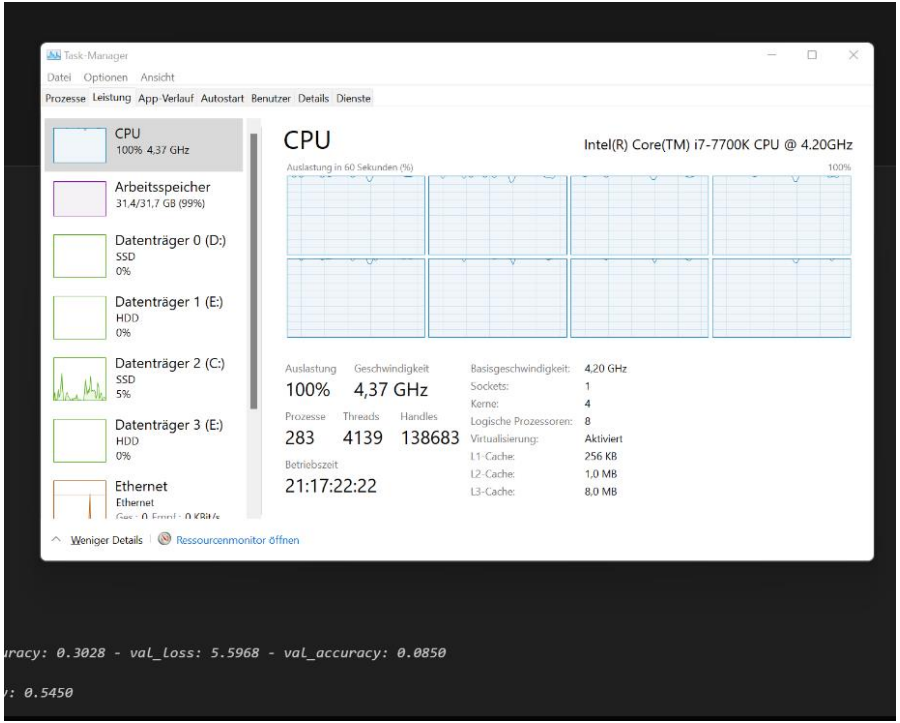
Daten in floats konvertiert und normalisiert

```
# in floats konvertieren
trainX = train_images.astype('float32')
testX = test_images.astype('float32')

# auf einen range von 0-1 normalisieren
trainX = trainX / 255.0
testX = testX / 255.0
```

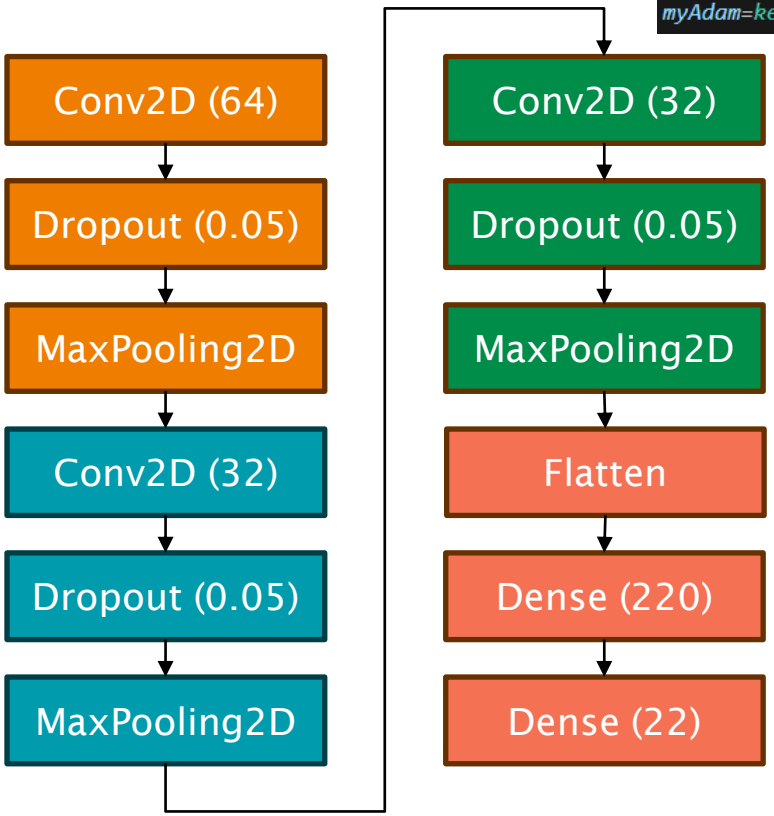
```
...  ---------------------------------------------------------------------------
     MemoryError                               Traceback (most recent call last)
     Cell In[8], line 15
          12 test_images = np.load("D:/KI_Plankton/test_arr.npy")
          14 # in floats konvertieren
     ---> 15 trainX = train_images.astype('float32')
          16 testX = test_images.astype('float32')
          18 # auf einen range von 0-1 normalisieren

     MemoryError: Unable to allocate 55.5 GiB for an array with shape (124114, 200, 200, 3) and data type float32
```

Probleme

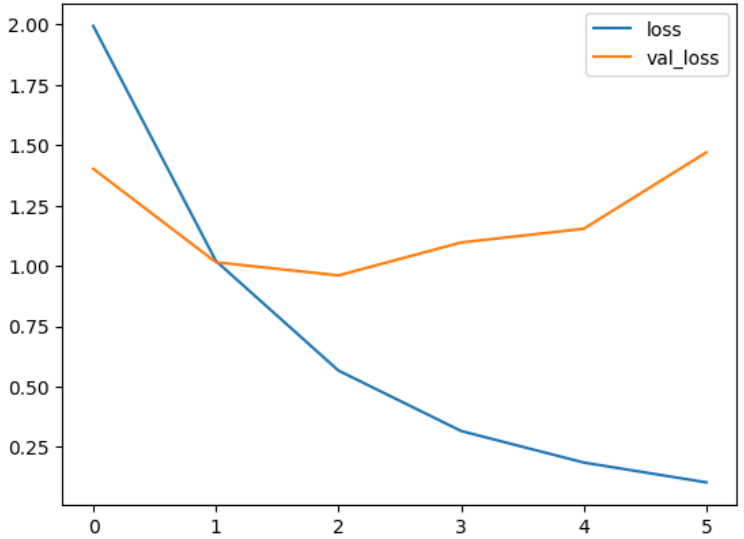

loss
val_loss
10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
88/88 [==========] - 991s 11s/step - loss: 0.7095 - accuracy: 0.7855 - val_loss: 1
modelCNN_cifar10.evaluate(testX, testY)
[8]
1m 2.8s
138/138 [==========] - 46s 327ms/step - loss: 11.1059 - accuracy: 0.0800
[11.105873107910156, 0.07999999821186066]


Task-Manager
Datei Optionen Ansicht
Prozesse Leistung App-Verlauf Autostart Benutzer Details Dienste
CPU
100% 4,37 GHz
Arbeitsspeicher
31,4/31,7 GB (99%)
Datenträger 0 (D:)
SSD
0%
Datenträger 1 (E:)
HDD
0%
Datenträger 2 (C:)
SSD
5%
Datenträger 3 (E:)
HDD
0%
Ethernet
Ethernet
CPU
Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
Auslastung in 60 Sekunden (%)
100%
Auslastung
100%
Geschwindigkeit
4,37 GHz
Prozesse
283
Threads
4139
Handles
138683
Betriebszeit
21:17:22:22
Basisgeschwindigkeit: 4,20 GHz
Sockets: 1
Kerne: 4
Logische Prozessoren: 8
Virtualisierung: Aktiviert
L1-Cache: 256 KB
L2-Cache: 1,0 MB
L3-Cache: 8,0 MB
Weniger Details
Ressourcenmonitor öffnen
uracy: 0.3028 - val_loss: 5.5968 - val_accuracy: 0.0850
: 0.5450

Neural Network



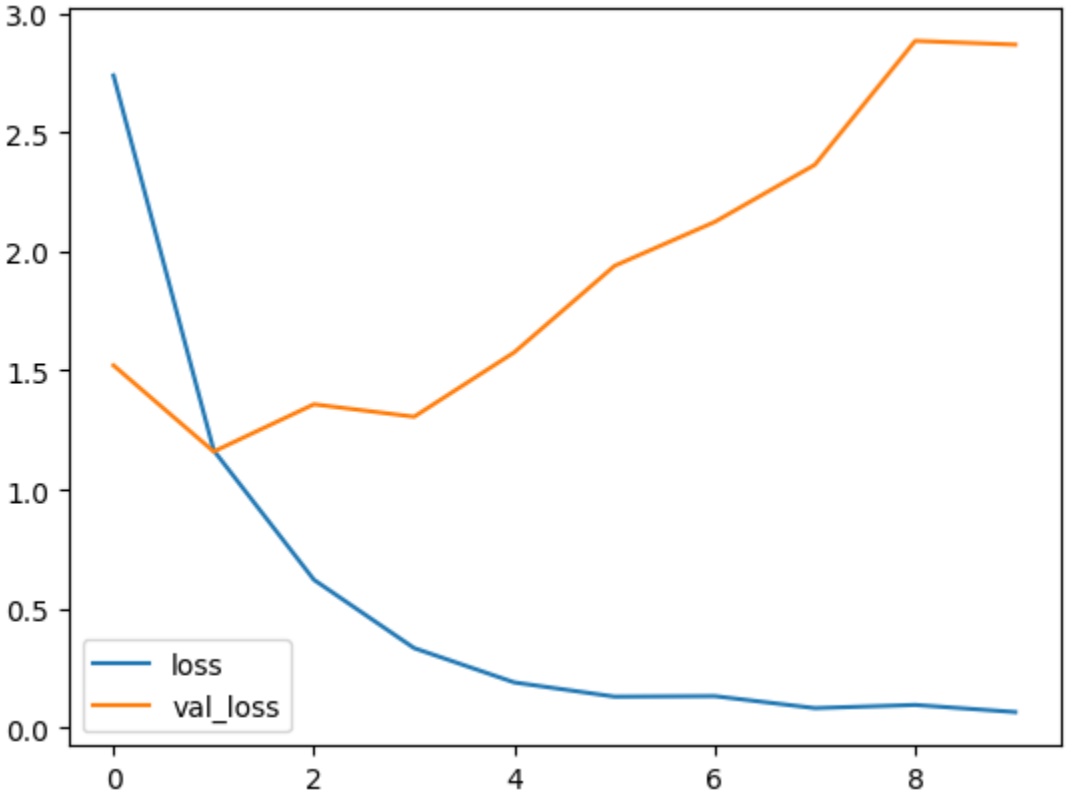

```
#Adam
myAdam=keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
```



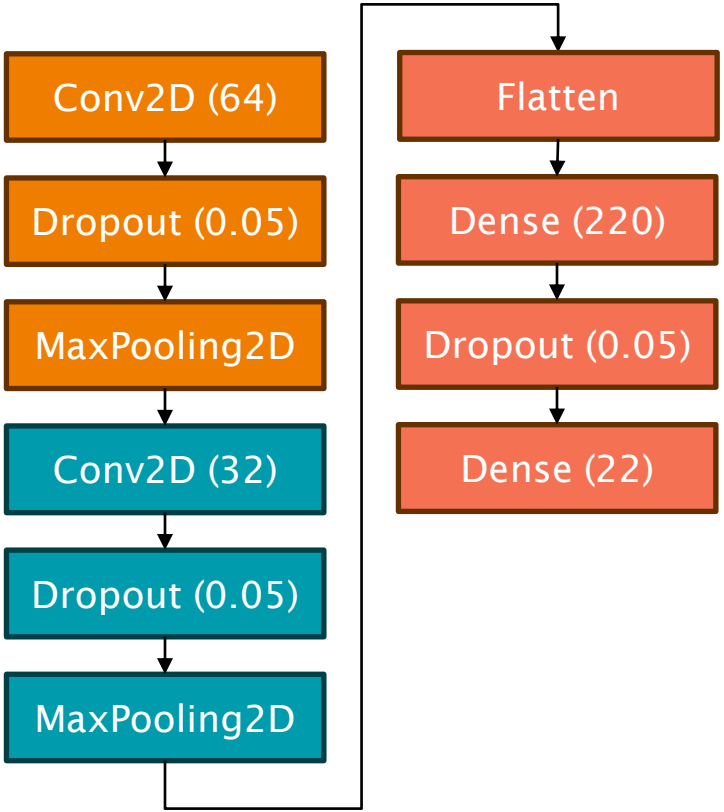

```
modelCNN_cifar10.evaluate(testX, testY)
```

```
138/138 [==============================] - 57s 412ms/step - loss: 1.4706 - accuracy: 0.7539
[1.470600962638855, 0.7538636326789856]
```

Neural Network

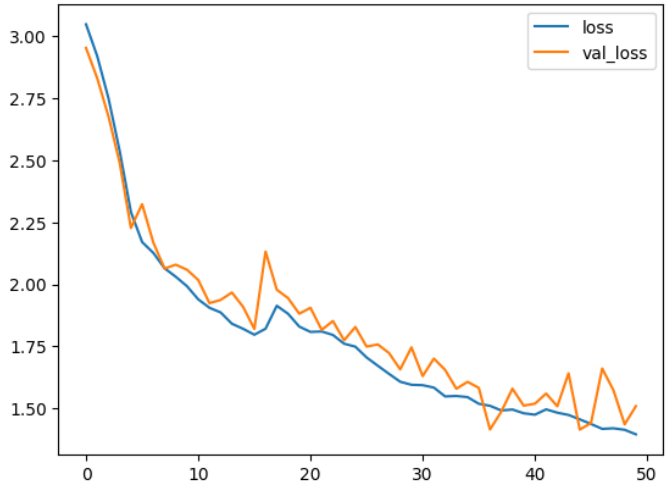


Neural Network



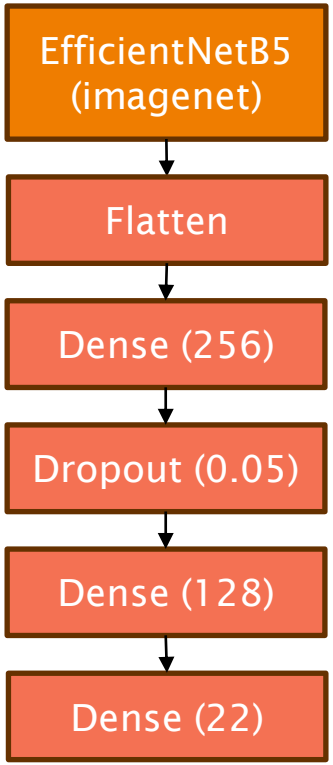

```
# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=10, # Rotation
        width_shift_range=0.2, # horizontaler Shift
        height_shift_range=0.2, # vertikaler Shift
        zoom_range=0.2, # zoom
        horizontal_flip=True) # flipping
```

```
#Stochastic gradient descent
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

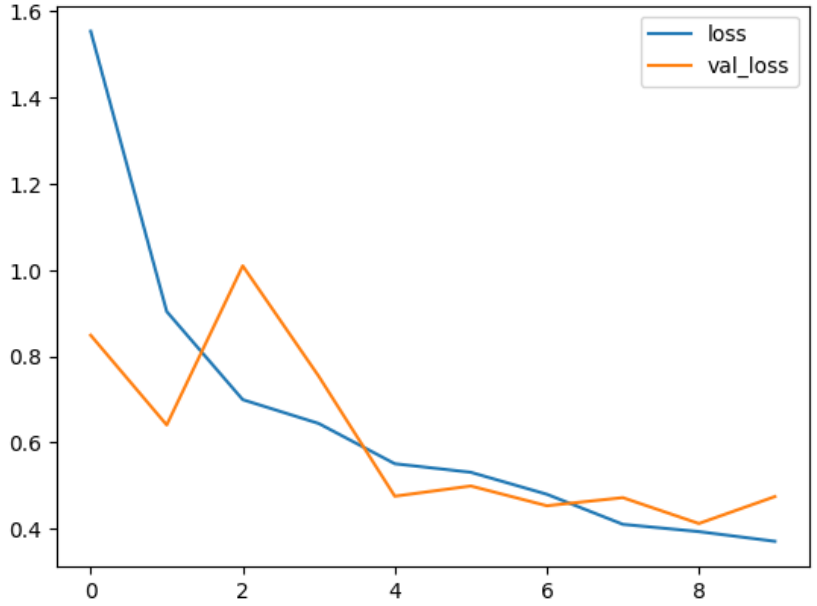



```
loss: 1.3954 - accuracy: 0.5514 - val_loss: 1.5090 - val_accuracy: 0.5195
```

Neural Network




```
#Adam
myAdam=keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
```




```
loss: 0.3712 - accuracy: 0.8852 - val_loss: 0.4749 - val_accuracy: 0.8714
```

https://github.com/xShadoow/ml-plankton