Crop Detection for Smart Farming

CS 646 Higher Level Computer Vision

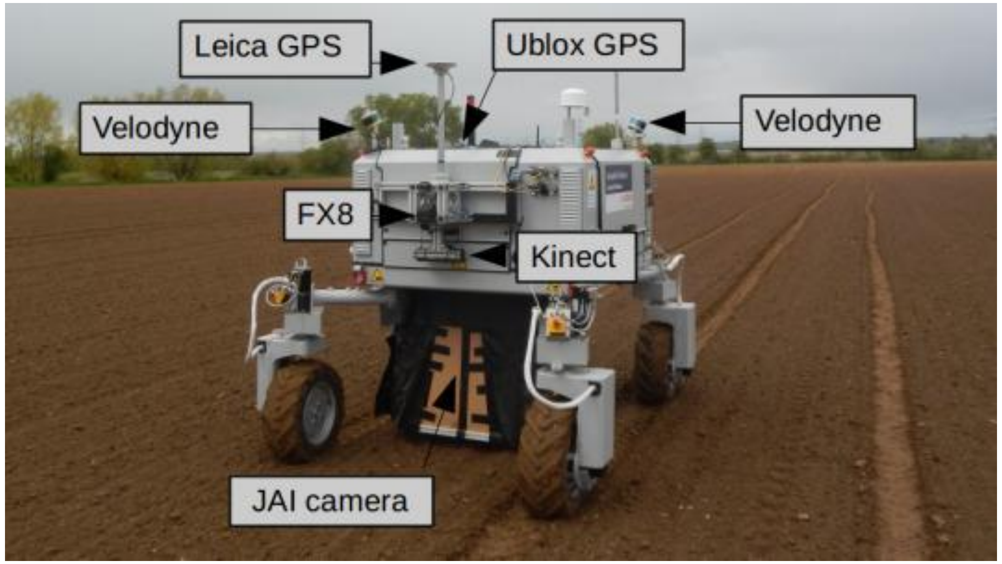
Esther Vogt

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Additional team members: Martin Böckling, David Probst, Fabio Westphal

Use Case in Smart Farming

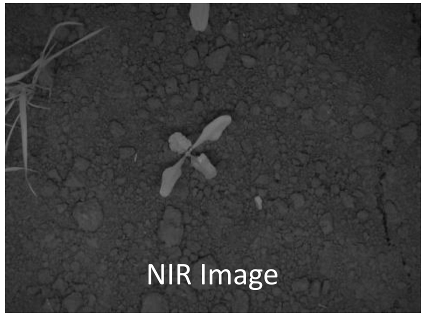
- Agricultural robots allow lower production costs & decreased need for manual labour
- Weed control: stop (noxious) weeds from competing with desired flora
- To apply targeted measures, weeds must be detected
- Robots carry cameras for weed detection

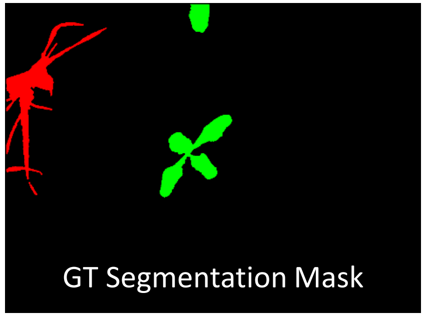


Data Set

- published by University of Bonn
- captured over period of three months in spring 2016
- robot carried 4-channel multi-spectral camera and RGB-D sensor
- used 11,552 / 12,340 of available labeled images
  - Input images: NIR (3 channels)
  - Segmentation mask: colorCleaned (3 classes)

Sample images





Data Availability

#images per category:

cka	nir	rgb	id	colorCleaned	iMapCleaned
CKA_160421	823	0	823	600	600
CKA_160426	306	306	306	296	296
CKA_160427	881	2	881	881	881
CKA_160428	300	0	300	300	300
CKA_160429	614	0	624	612	602
CKA_160502	302	0	302	302	302
CKA_160503	1570	288	1570	1556	1556
CKA_160504	301	0	301	301	301
CKA_160505	963	0	963	963	963
CKA_160506	301	0	301	301	301
CKA_160509	912	0	912	912	912
CKA_160510	867	0	869	869	867
CKA_160511	401	1	401	289	289
CKA_160512	304	0	304	304	304
CKA_160513	301	0	332	319	314
CKA_160517	307	0	315	315	307
CKA_160518	305	0	305	305	305
CKA_160523	2148	3	2281	1949	2197
CKA_160527	300	0	300	300	300
CKA_weeds	512	0	522	0	505

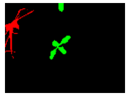
Problem: sparse RGB image availability

Data Understanding & Preprocessing

UNIVERSITY OF MANNHEIM
School of Business Informatics and Mathematics

Data Understanding

- Class ratio (avg):



Background: 97.8%

Crop: 2.0%

Weed: 0.4%

Data Preprocessing

- Input images: Resizing, CenterCrop, **Normalization**

```
tensor_image_nir = self.transform_img(image_nir)
```

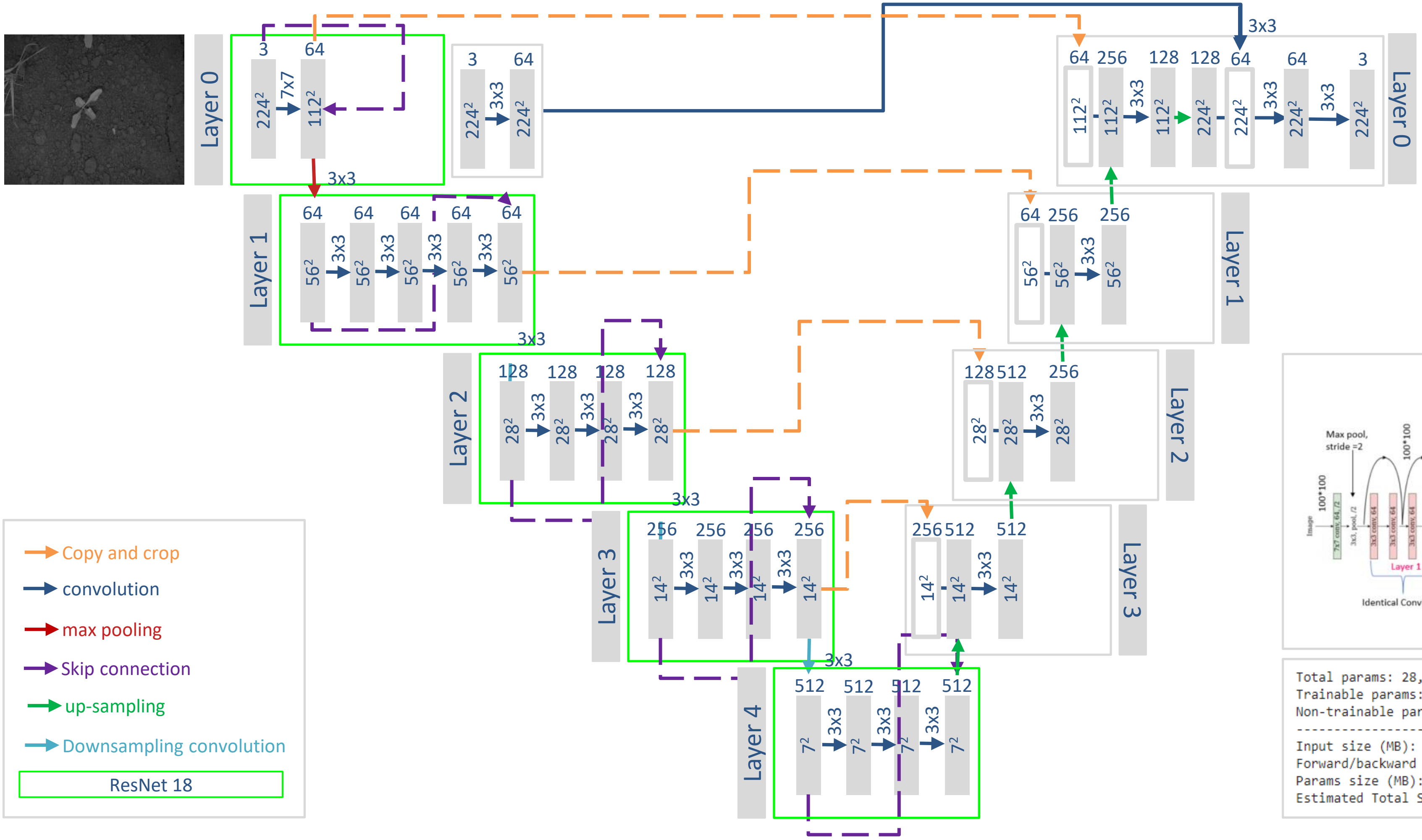
```
preprocess_img = transforms.Compose([
  transforms.Resize(256),
  transforms.CenterCrop(224),
  transforms.ToTensor(),
  transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

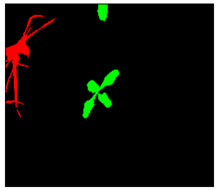
- GT Segmentation masks: Resizing, CenterCrop, **Transformation to binary class channels**

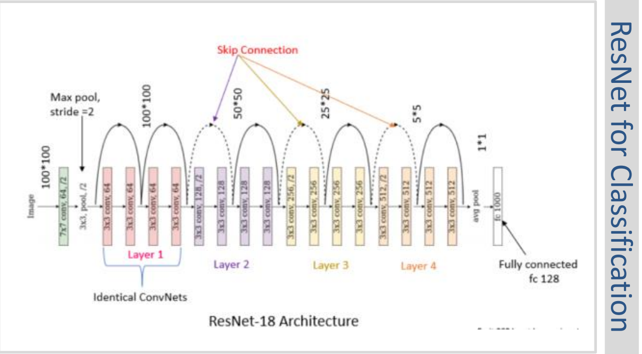
```
# (1) apply preprocess-mask: resize+centercrop  (note: this changes color values !)
image_label = preprocess_mask(image_label)

# (2) convert PIL -> np -> torch
label_torch = torch.from_numpy(np.array(image_label))

# (3a) convert values to one-hot: after resize and centercrop, some pixels at class
#      borders have values other than 0 and 255 to encode some degree of uncertainty
#   -> approach: everything below 128 as 0, everything above as 1
label_torch[(label_torch<=128) & (label_torch!=0)] = 0
label_torch[(label_torch>128)] = 1

# (3b) convert RGB-channels into segmentation mask with class channels
#   -> this will also give one-hot to background
one_hot_label = mask_to_onehot(label_torch, palette)

# (4) swap dimensions: #classes x h x w -> to fit input dims
one_hot_label = one_hot_label.long().permute(2,0,1)
```

```
preprocess_mask = transforms.Compose([
  transforms.Resize(256),
  transforms.CenterCrop(224),
])
```

Model Architecture – ResNet (UNet)








```
Total params: 28,976,451
Trainable params: 28,976,451
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 416.50
Params size (MB): 110.54
Estimated Total Size (MB): 527.61
```

params

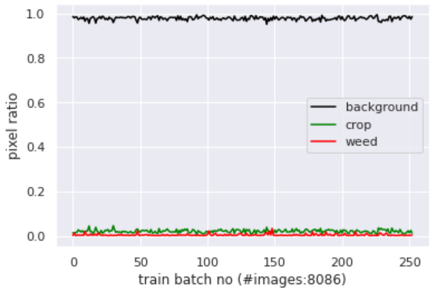
Model Training

Approach

- Loss Metric: 1 – mIoU vs. 1-weighted mIoU
- Performance Metric: mIoU vs. weighted mIoU
- # images: 11,552
- Train-Test Split:
- 0.70 – training set
- 0.15 – validation set
- 0.15 – test set
- # epochs: 30
- Batch size: 32
- Activation: ReLu

Model Training

Lessons Learned – mean IoU vs. weighted mean IoU



Effective Number of Samples (ENS)

$$w_{n,c} = \frac{1}{E_{n_c}}$$

$$E_{n_c} = \frac{1-\beta^{n_c}}{1-\beta}$$

where n_c is the Number of Samples in Class c and E_{n_c} represents the Effective Number of Samples

Sample batch of size 15

	# pixels			IoU loss			IoU loss (avg)
batch no	bg	crop	weed	bg	crop	weed	avg
1	50.006	130	40	0,0070	1,0000	0,9989	0,6686
2	49.570	554	52	0,0164	0,9983	0,9986	0,6711
3	50.138	11	27	0,0054	0,9965	0,9992	0,6670
4	49.463	688	25	0,0169	1,0000	1,0000	0,6723
5	49.446	634	96	0,0182	1,0000	1,0000	0,6727
6	49.050	1.061	65	0,0248	1,0000	1,0000	0,6749
7	49.864	312	0	0,0092	1,0000	1,0000	0,6697
8	49.762	213	201	0,0128	1,0000	0,9999	0,6709
9	50.021	135	20	0,0061	0,9987	1,0000	0,6683
10	49.699	463	14	0,0129	1,0000	1,0000	0,6710
11	49.602	574	0	0,0140	1,0000	1,0000	0,6713
12	49.812	364	0	0,0105	1,0000	1,0000	0,6702
13	49.693	433	50	0,0123	1,0000	0,9993	0,6705
14	50.037	122	17	0,0065	0,9974	0,9964	0,6668
15	50.018	145	13	0,0061	1,0000	1,0000	0,6687
						avg	0,6703

$\beta = 0,99$

weights (ENS)			wIoU loss (ENS)			wIoU loss (avg)
bg	crop	weed	bg	crop	weed	avg
0,19	0,25	0,56	0,00	0,2543	0,5596	0,8152
0,22	0,23	0,55	0,00	0,2247	0,5500	0,7784
0,07	0,65	0,29	0,00	0,6450	0,2848	0,9302
0,15	0,15	0,69	0,00	0,1540	0,6922	0,8488
0,28	0,28	0,45	0,01	0,2769	0,4466	0,7286
0,24	0,24	0,51	0,01	0,2448	0,5104	0,7613
0,49	0,51	0,00	0,00	0,5111	0,0000	0,5156
0,30	0,34	0,35	0,00	0,3449	0,3508	0,6996
0,13	0,17	0,70	0,00	0,1716	0,7006	0,8730
0,10	0,10	0,79	0,00	0,1049	0,7913	0,8975
0,50	0,50	0,00	0,01	0,5008	0,0000	0,5078
0,49	0,51	0,00	0,01	0,5065	0,0000	0,5117
0,22	0,22	0,56	0,00	0,2229	0,5567	0,7823
0,11	0,16	0,72	0,00	0,1607	0,7223	0,8838
0,10	0,12	0,78	0,00	0,1245	0,7799	0,9051
					avg	0,7626

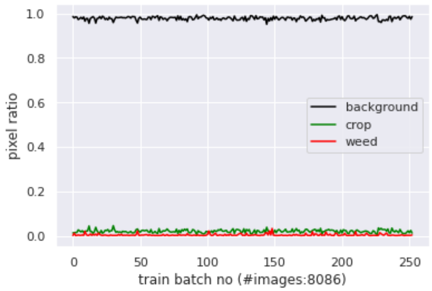
$\beta = 0,999$

weights (ENS)			wIoU loss (ENS)			wIoU loss (avg)
bg	crop	weed	bg	crop	weed	avg
0,03	0,24	0,73	0,00	0,2364	0,7340	0,9706
0,04	0,10	0,85	0,00	0,1017	0,8536	0,9560
0,01	0,70	0,29	0,00	0,7010	0,2886	0,9897
0,02	0,05	0,93	0,00	0,0462	0,9308	0,9774
0,07	0,15	0,78	0,00	0,1515	0,7772	0,9301
0,05	0,08	0,86	0,00	0,0830	0,8626	0,9470
0,21	0,79	0,00	0,00	0,7886	0,0000	0,7905
0,09	0,45	0,47	0,00	0,4453	0,4692	0,9156
0,02	0,13	0,85	0,00	0,1331	0,8499	0,9831
0,01	0,04	0,95	0,00	0,0357	0,9511	0,9869
0,30	0,70	0,00	0,00	0,6959	0,0000	0,7002
0,23	0,77	0,00	0,00	0,7661	0,0000	0,7686
0,04	0,12	0,84	0,00	0,1169	0,8415	0,9589
0,01	0,13	0,86	0,00	0,1258	0,8562	0,9821
0,01	0,09	0,90	0,00	0,0863	0,9020	0,9884
					avg	0,9230

Source:
- Cui, Y., Jia, M., Lin, T. Y., Song, Y., & Belongie, S. (2019). Class-balanced loss based on effective number of samples. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9268-9277).
- https://medium.com/gumgum-tech/handling-class-imbalance-by-introducing-sample-weighting-in-the-loss-function-3bdebd8203b4

Model Evaluation

Lessons Learned – mean IoU vs. weighted mean IoU



Effective Number of Samples (ENS)

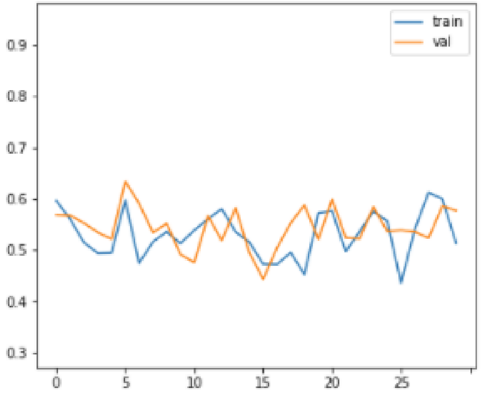
$$w_{n,c} = \frac{1}{E_{n_c}}$$

$$E_{n_c} = \frac{1-\beta^{n_c}}{1-\beta}$$

where n_c is the Number of Samples in Class c
and E_{n_c} represents the Effective Number of Samples

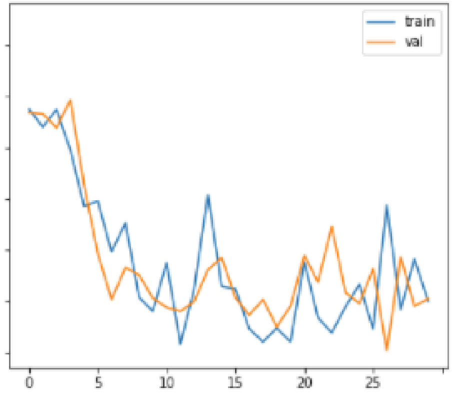
Loss curve train vs. validation (lr=1e-4)

IoU loss: 0.44



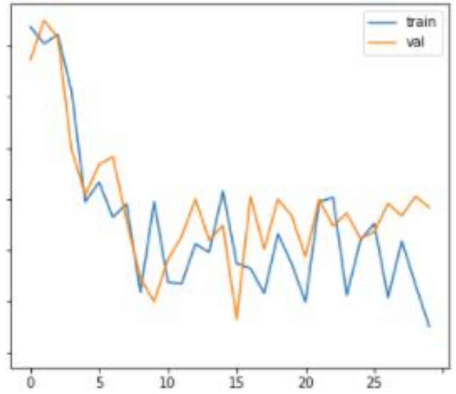
$\beta = 0{,}99$

weighted IoU loss: 0.30



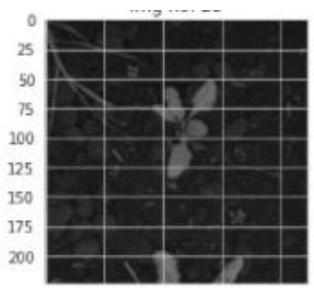
$\beta = 0{,}999$

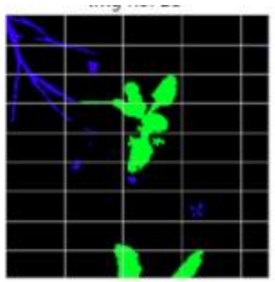
weighted IoU loss: 0.36



Model Evaluation

Lessons Learned – mean IoU vs. weighted mean IoU





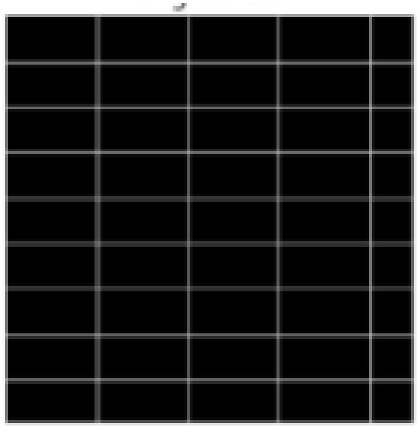
Effective Number of Samples (ENS)

$$w_{n,c} = \frac{1}{E_{n_c}}$$

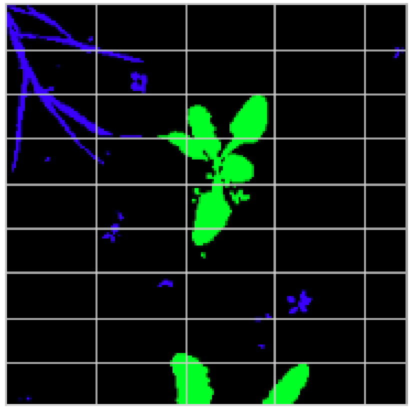
$$E_{n_c} = \frac{1-\beta^{n_c}}{1-\beta}$$

where n_c is the Number of Samples in Class c
and E_{n_c} represents the Effective Number of Samples

Predicted segmentation mask (lr=1e-4)



$\beta = 0{,}99$



$\beta = 0{,}999$