SCML Seminar #3

Hannes Liik
26.09.2019

Playing with CAMVID (1)

What happens when we finetune the head of an untrained model?

```
learn = unet_learner(data, models.resnet34, metrics=metrics, wd=wd, pretrained=False)
learn.freeze() # Just to be sure
```

```
learn = unet_learner(data, models.resnet34, metrics=metrics, wd=wd, pretrained=False)
learn.freeze() # Just to be sure
```

70% with
random features

```
lr=3e-3
```

```
learn.fit_one_cycle(10, slice(lr), pct_start=0.9)
```

epoch	train_loss	valid_loss	acc_camvid	time
0	1.382325	1.406335	0.598760	01:01
1	1.155398	1.293682	0.629446	01:00
2	1.075509	1.125265	0.592909	01:01
3	1.047244	1.183537	0.659575	01:01
4	1.022191	0.985601	0.700472	01:01
5	1.013113	0.998961	0.679257	01:01
6	1.003547	1.017310	0.662229	01:00
7	1.300187	1.279307	0.587653	00:58
8	1.061010	1.107628	0.651201	01:00
9	0.954468	1.040581	0.671118	01:01

Playing with CAMVID (2) Data augmentation?

In [32]:
```
print("Standard transforms:")
print(get_transforms())
```

```
Standard transforms:
([RandTransform(tfm=TfmCrop (crop_pad), kwargs={'row_pct': (0, 1), 'col_pct': (0, 1), 'padding_mode': 'reflectio
n'}, p=1.0, resolved={}, do_run=True, is_random=True, use_on_y=True), RandTransform(tfm=TfmPixel (flip_lr), kwargs
={}, p=0.5, resolved={}, do_run=True, is_random=True, use_on_y=True), RandTransform(tfm=TfmCoord (symmetric_warp),
kwargs={'magnitude': (-0.2, 0.2)}, p=0.75, resolved={}, do_run=True, is_random=True, use_on_y=True), RandTransform
(tfm=TfmAffine (rotate), kwargs={'degrees': (-10.0, 10.0)}, p=0.75, resolved={}, do_run=True, is_random=True, use_
on_y=True), RandTransform(tfm=TfmAffine (zoom), kwargs={'scale': (1.0, 1.1), 'row_pct': (0, 1), 'col_pct': (0,
1)}, p=0.75, resolved={}, do_run=True, is_random=True, use_on_y=True), RandTransform(tfm=TfmLighting (brightness),
kwargs={'change': (0.4, 0.6)}, p=0.75, resolved={}, do_run=True, is_random=True, use_on_y=True), RandTransform(tfm
=TfmLighting (contrast), kwargs={'scale': (0.8, 1.25)}, p=0.75, resolved={}, do_run=True, is_random=True, use_on_y
=True)], [RandTransform(tfm=TfmCrop (crop_pad), kwargs={}, p=1.0, resolved={}, do_run=True, is_random=True, use_on
_y=True)])
```

In [33]:
```
tfms = get_transforms(do_flip=False, max_rotate=0, max_zoom=1, max_lighting=0, max_warp=0, p_affine=0, p_lighting=0
print("Minimal transforms")
print(tfms)
```

```
Minimal transforms
([RandTransform(tfm=TfmCrop (crop_pad), kwargs={'row_pct': (0, 1), 'col_pct': (0, 1), 'padding_mode': 'reflectio
n'}, p=1.0, resolved={}, do_run=True, is_random=True, use_on_y=True)], [RandTransform(tfm=TfmCrop (crop_pad), kwar
gs={}, p=1.0, resolved={}, do_run=True, is_random=True, use_on_y=True)])
```

In [34]:
```
data = (src.transform(tfms, size=size, tfm_y=True)
        .databunch(bs=bs)
        .normalize(imagenet_stats))
```

91.5% acc

The data augmentation did surprisingly little

```
In [42]: learn.unfreeze()

In [43]: lrs = slice(lr/400,lr/4)

In [44]: learn.fit_one_cycle(12, lrs, pct_start=0.8)
```

epoch	train_loss	valid_loss	acc_camvid	time
0	0.298258	0.331086	0.898335	01:02
1	0.286851	0.324747	0.901298	01:05
2	0.281320	0.320566	0.903245	01:06
3	0.280698	0.326616	0.904825	01:06
4	0.263327	0.321662	0.907090	01:05
5	0.255840	0.321796	0.906381	01:02
6	0.245424	0.370866	0.893723	01:04
7	0.238833	0.302455	0.912225	01:05
8	0.213995	0.313009	0.916014	01:05
9	0.199943	0.336384	0.910490	01:05
10	0.182788	0.341570	0.912375	01:05
11	0.166621	0.334465	0.915304	01:05

Batch size and crop size

Rethinking Atrous Convolution for Semantic Image Segmentation

Liang-Chieh Chen George Papandreou Florian Schroff Hartwig Adam
Google Inc.
{lcchen, gpapan, fschroff, hadam}@google.com

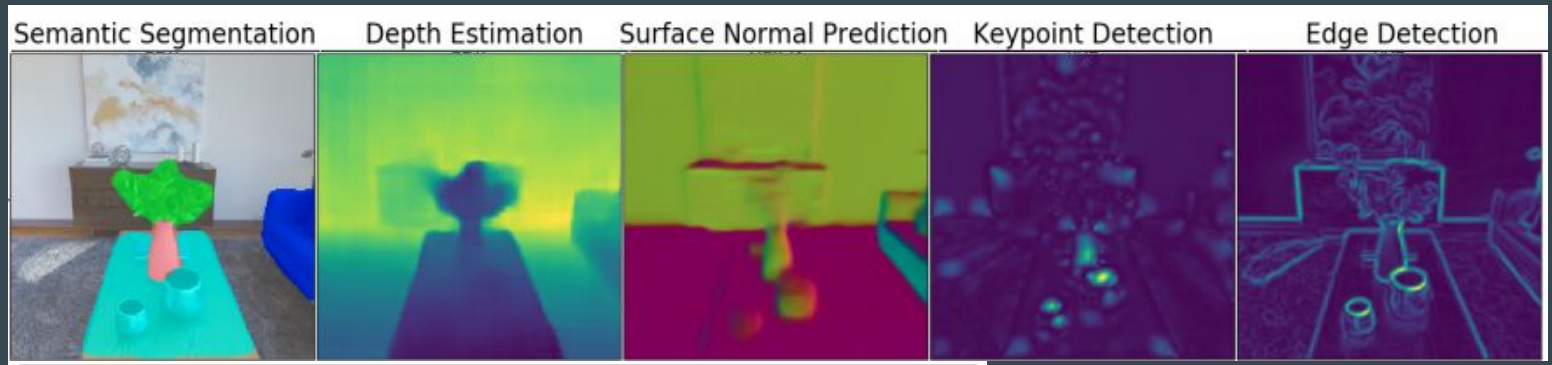
Crop Size	UL	BN	mIOU
513	✓	✓	77.21
513	✓		75.95
513		✓	76.01
321		✓	67.22

Table 8. Effect of hyper-parameters during training on PASCAL VOC 2012 *val* set at *output_stride=16*. **UL**: Upsampling Logits. **BN**: Fine-tuning batch normalization.

batch size	mIOU
4	64.43
8	75.76
12	76.49
16	77.21

Table 9. Effect of batch size on PASCAL VOC 2012 *val* set. We employ *output_stride=16* during both training and evaluation. Large batch size is required while training the model with fine-tuning the batch normalization parameters.

Multi task learning



		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	-5.41%	-11.29%	-4.32%	-34.64%	-13.92%
	Depth	4.17%	–	-3.55%	3.49%	3.76%	1.97%
	Normals	8.50%	2.48%	–	1.37%	12.33%	6.17%
	Keypoints	4.82%	1.38%	-0.02%	–	-5.26%	0.23%
	Edges	3.07%	-0.92%	-4.42%	1.37%	–	-0.23%
	Average	5.14%	-0.62%	-4.82%	0.48%	-5.95%	-1.15%

Table 1. **The first-order multi-task learning relationships between tasks.** The table lists the performance of every task when trained as a pair with every other task. For instance, when Depth is trained with SemSeg, SemSeg performs 4.17% better than when SemSeg is trained alone on a half-sized network.

Which Tasks Should Be Learned Together in Multi-task Learning?

Trevor Standley[1] Amir R. Zamir[1,2] Dawn Chen[3] Leonidas Guibas[1] Jitendra Malik[2] Silvio Savarese[1]

[1]Stanford University [2]The University of California, Berkeley [3]Google Inc.

http://taskgrouping.stanford.edu/

Network	Pix acc.	Mean acc	Mean IU	fw IU
With depth	**0.89**	**0.76**	**0.61**	**0.81**
No depth	0.65	0.71	0.47	0.55

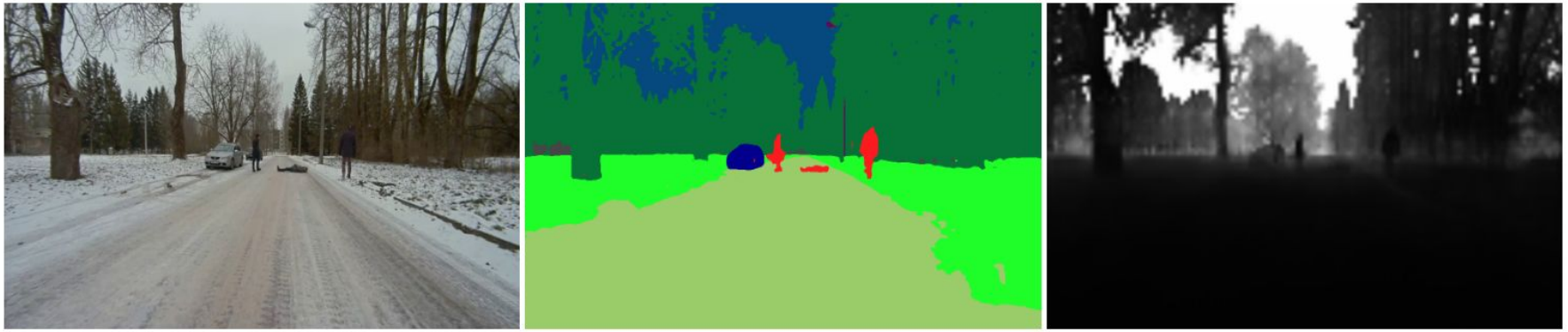


Figure 2. Prediction from segmentation + depth network

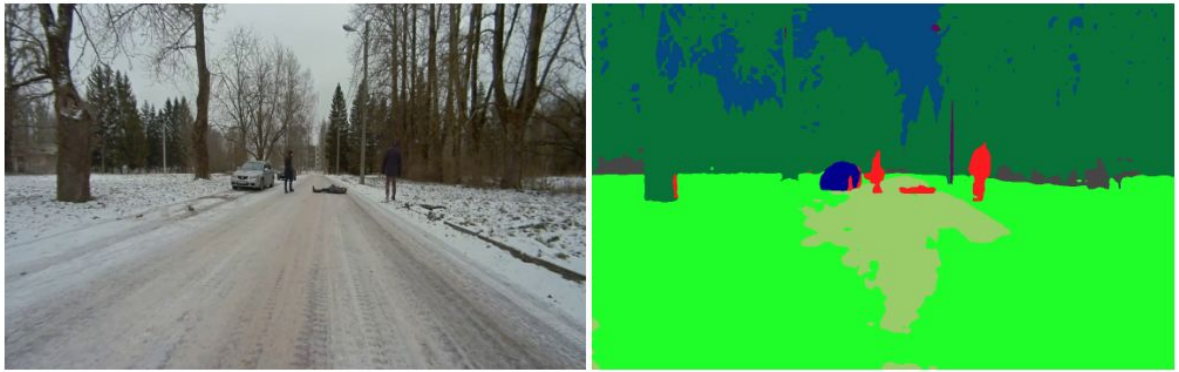
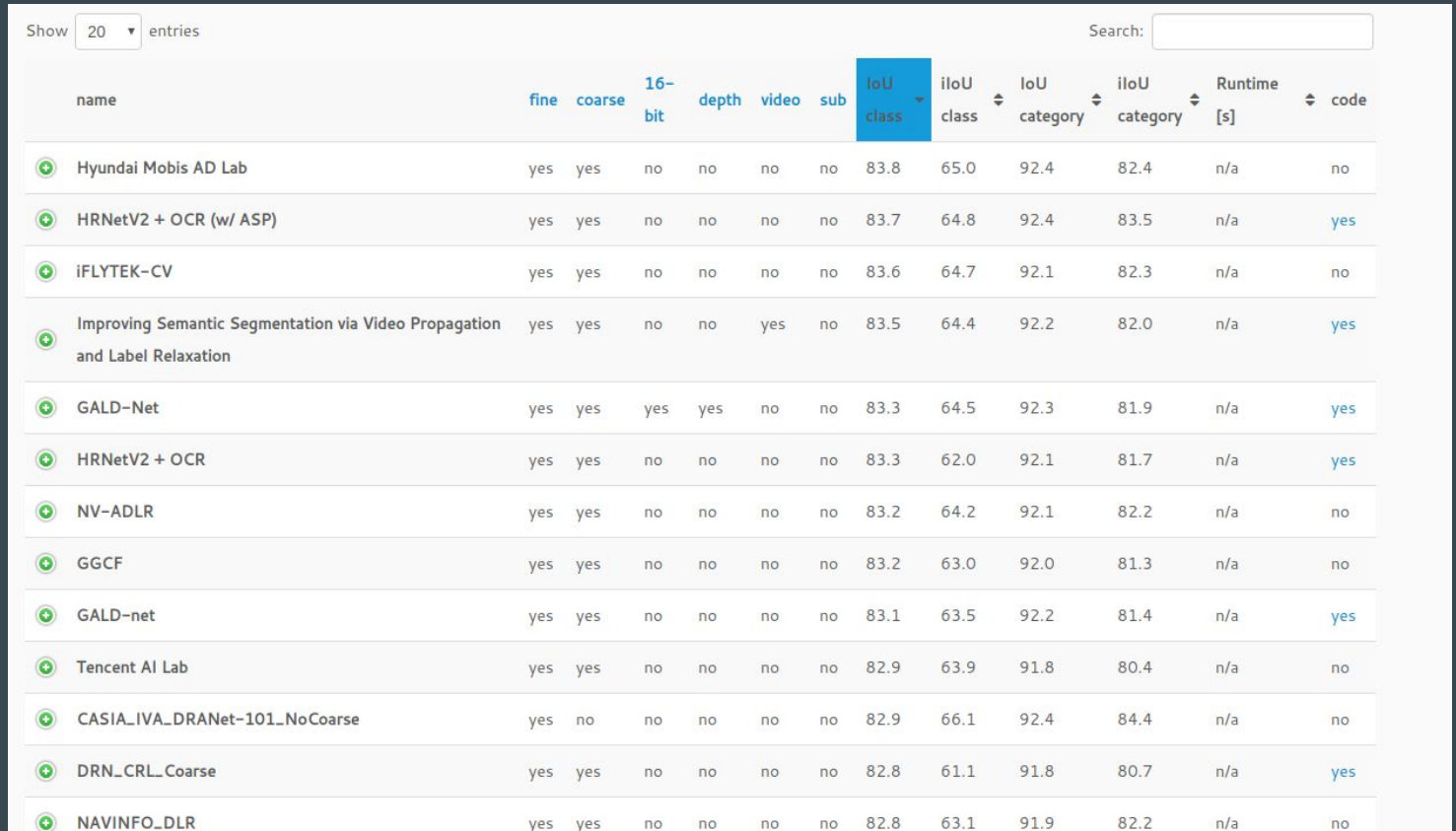


Figure 3. Prediction from segmentation-only network

Where to find segmentation networks - Cityscapes leaderboard



Show 20 entries

Search:

name	fine	coarse	16-bit	depth	video	sub	IoU class	iIoU class	IoU category	iIoU category	Runtime [s]	code
Hyundai Mobis AD Lab	yes	yes	no	no	no	no	83.8	65.0	92.4	82.4	n/a	no
HRNetV2 + OCR (w/ ASP)	yes	yes	no	no	no	no	83.7	64.8	92.4	83.5	n/a	yes
iFLYTEK-CV	yes	yes	no	no	no	no	83.6	64.7	92.1	82.3	n/a	no
Improving Semantic Segmentation via Video Propagation and Label Relaxation	yes	yes	no	no	yes	no	83.5	64.4	92.2	82.0	n/a	yes
GALD-Net	yes	yes	yes	yes	no	no	83.3	64.5	92.3	81.9	n/a	yes
HRNetV2 + OCR	yes	yes	no	no	no	no	83.3	62.0	92.1	81.7	n/a	yes
NV-ADLR	yes	yes	no	no	no	no	83.2	64.2	92.1	82.2	n/a	no
GGCF	yes	yes	no	no	no	no	83.2	63.0	92.0	81.3	n/a	no
GALD-net	yes	yes	no	no	no	no	83.1	63.5	92.2	81.4	n/a	yes
Tencent AI Lab	yes	yes	no	no	no	no	82.9	63.9	91.8	80.4	n/a	no
CASIA_IVA_DRANet-101_NoCoarse	yes	no	no	no	no	no	82.9	66.1	92.4	84.4	n/a	no
DRN_CRL_Coarse	yes	yes	no	no	no	no	82.8	61.1	91.8	80.7	n/a	yes
NAVINFO_DLR	yes	yes	no	no	no	no	82.8	63.1	91.9	82.2	n/a	no

More remarks

- More classes means more supervision signal. Merge classes together at inference time.
- Segmentation labels are dense and less images are required than in classification/detection