



International
Centre for
Radio
Astronomy
Research

Looking for change? Roll the
Dice and demand Attention

Foivos I. Diakogiannis, Francois Waldner, Peter Caccetta 2020
(under review)

<https://github.com/feevos/ceecnet>



Government of Western Australia
Department of the Premier and Cabinet
Office of Science

Change detection: what is it?



Looking for change? Roll the Dice and demand Attention

Foivos I. Diakogiannis^{a,b,1}, François Waldner^c, Peter Caccetta^b

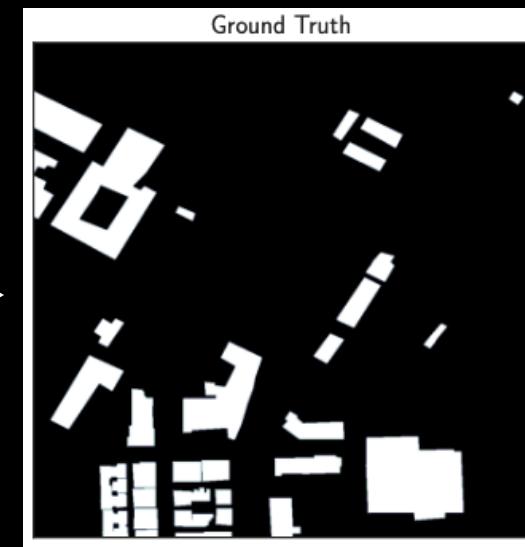
^aICRAR, the University of Western Australia

^bData61, CSIRO, Floreat WA

^cCSIRO Agriculture & Food, St Lucia, QLD, Australia

under review

Compare (DL)

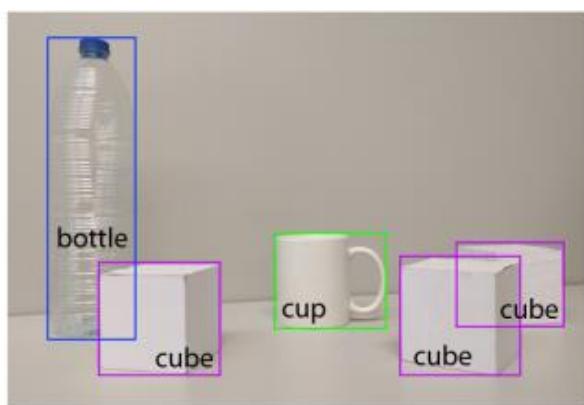


Change detection is a double input – single output **semantic segmentation** problem (with some peculiarities)

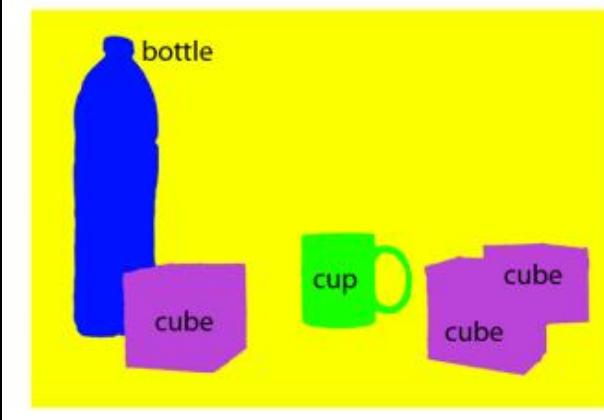
A necessary parenthesis: semantic segmentation definition + lessons learned



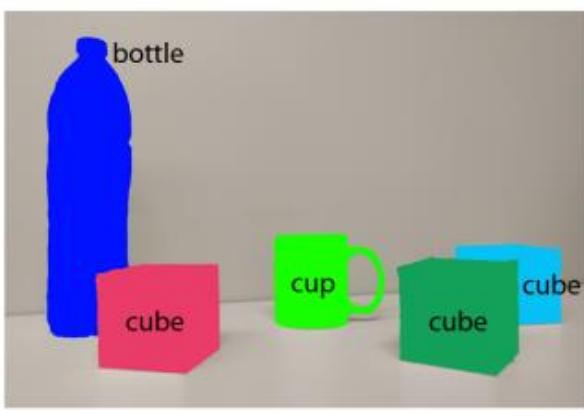
(a) Image classification



(b) Object localization



(c) Semantic segmentation



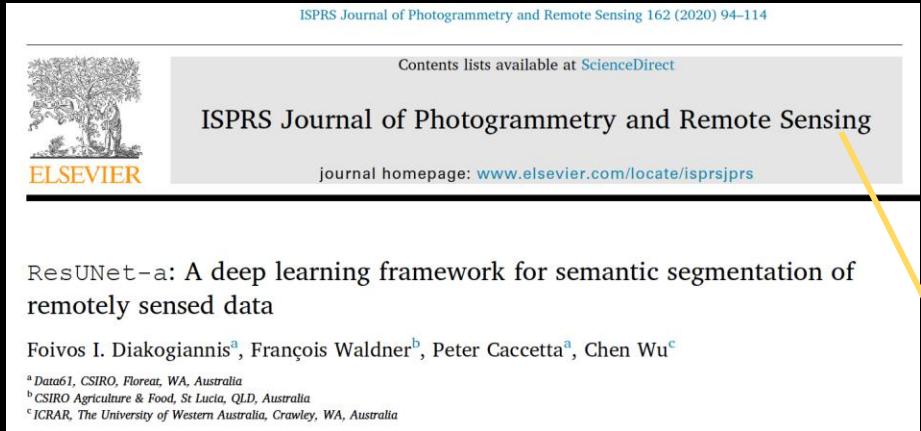
(d) Instance segmentation

A Review on Deep Learning Techniques Applied to Semantic Segmentation

A. Garcia-Garcia, S. Orts-Escalano, S.O. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez

<https://arxiv.org/abs/1704.06857>

Semantic Segmentation: lessons learned



<https://github.com/feevos/resuneta>

1. Atrous convolutions (the good and the bad: can we do better?)
2. Importance of context information
3. Analysis of Dice family of loss functions: can we do better?
4. “Causal” (conditioned) **multitasking**
5. Emphasis on boundaries

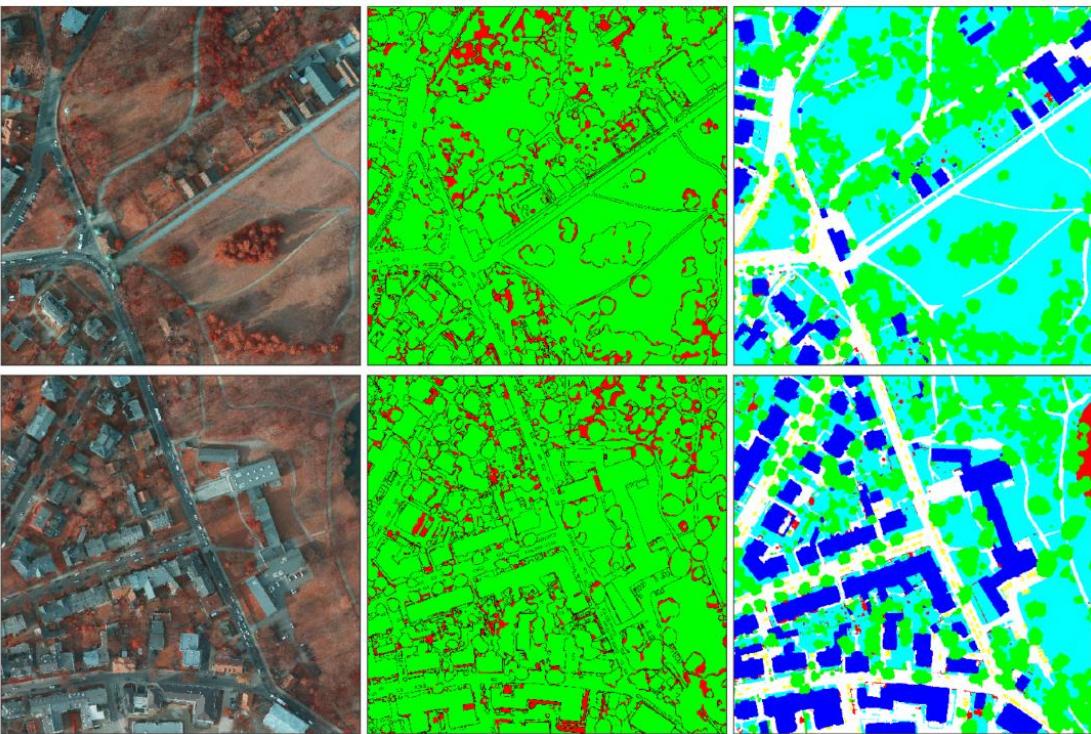


Table 5

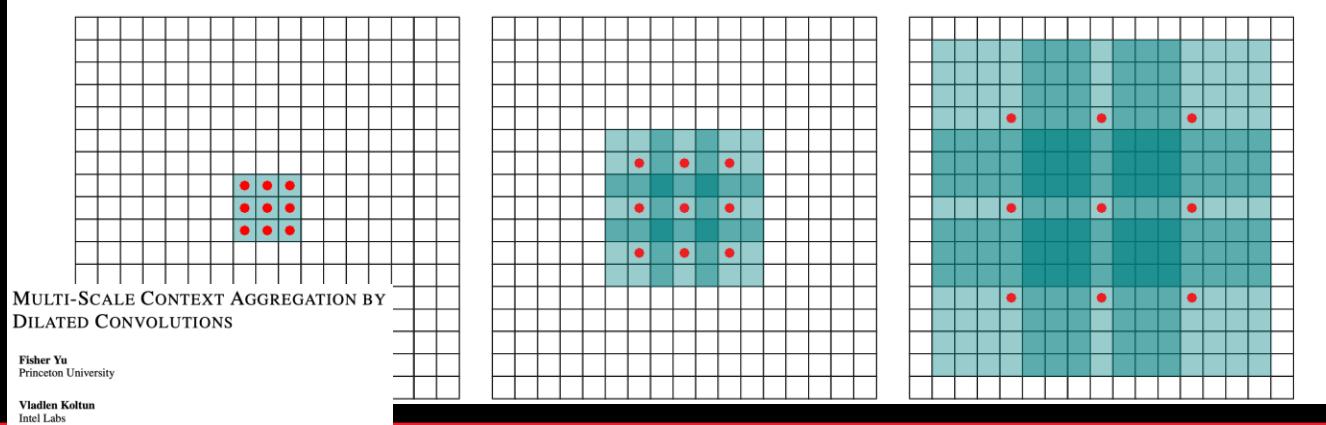
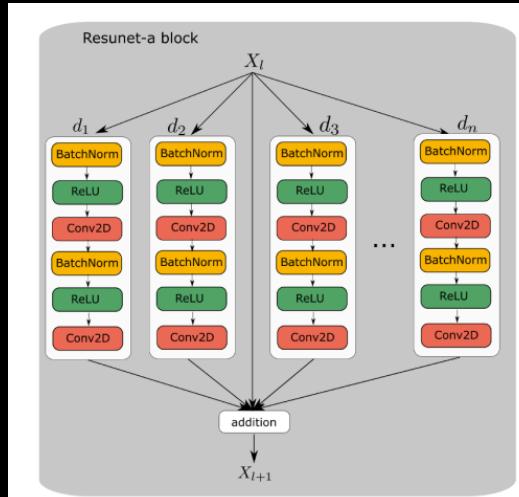
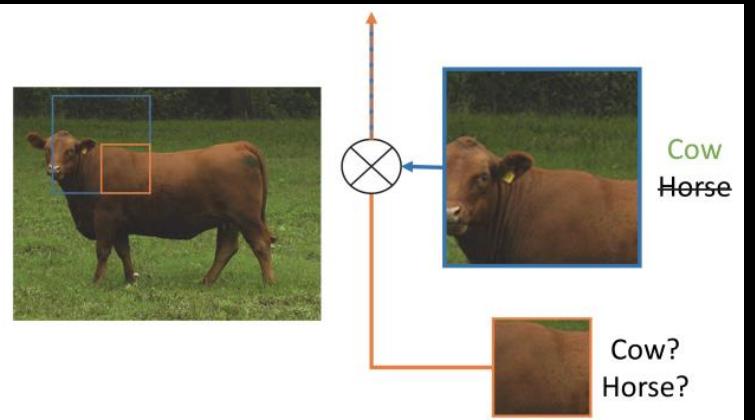
Potsdam comparison of results (based on per class F1 score) with other authors. Best values are marked with bold, second best values are underlined, third best values are in square brackets. Models trained with FoV × 1 were trained on 256x256 patches extracted from the original resolution images.

Methods	ImSurface	Building	LowVeg	Tree	Car	Avg. F1	OA
UZ_1 (Volpi and Tula, 2017)	89.3	95.4	81.8	80.5	86.5	86.7	85.8
RIT_L7 (Liu et al., 2017b)	91.2	94.6	85.1	85.1	92.8	89.8	88.4
RIT_4 (Piramanyagam et al., 2013)	92.6	<u>97.0</u>	86.9	87.4	95.2	91.8	90.3
DST_5 (Sherrah, 2016)	92.5	96.4	86.7	<u>88.8</u>	94.7	91.7	90.3
CAS_Y3 (ISPRS)	92.2	95.7	87.2	87.6	95.6	91.7	90.1
CASIA2 (Liu et al., 2018)	<u>93.3</u>	<u>97.0</u>	[87.7]	[88.4]	<u>96.2</u>	<u>92.5</u>	<u>91.1</u>
DPN_MFFL (Pan et al., 2018b)	92.4	[96.4]	<u>87.8</u>	88.0	95.7	92.1	90.4
HSN + OI + WBP (Liu et al., 2017a)	91.8	95.7	84.4	79.6	88.3	87.9	89.4
ResUNet-a d6 cmtsk (FoV × 1)	[93.0]	97.2	87.5	[88.4]	[96.1]	[92.4]	[91.0]
ResUNet-a d7v2 cmtsk (FoV × 1)	93.5	97.2	<u>88.2</u>	<u>89.2</u>	96.4	92.9	91.5

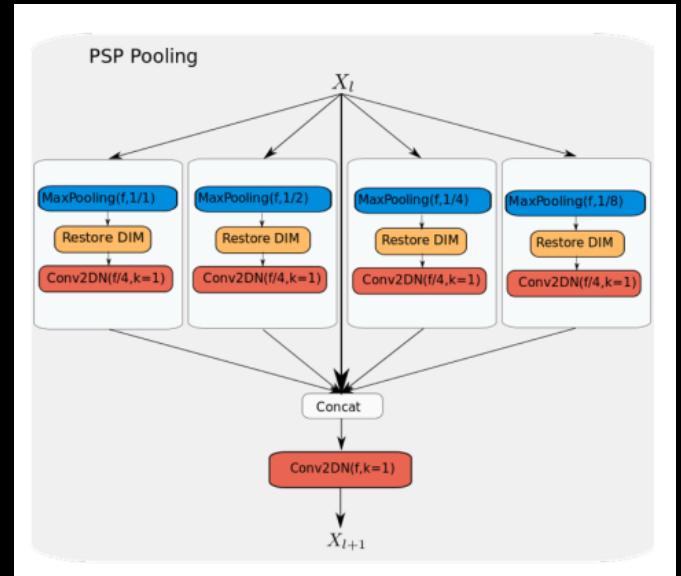
Semantic Segmentation: lessons learned

atrous convolutions:

- increase receptive field (dropout-like regularization???)
- They do not emphasize specific regions of images (attention does!)

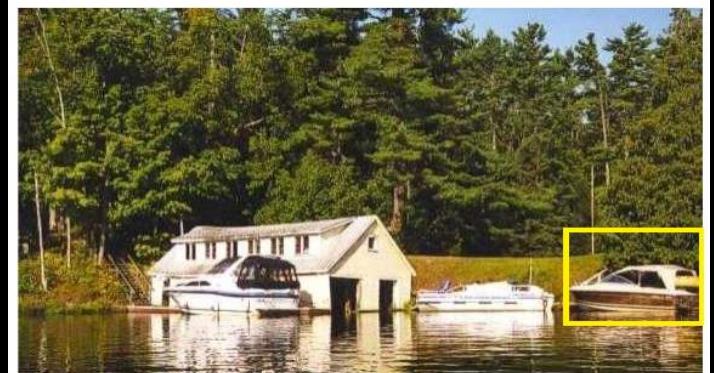


PSPPooling: context information



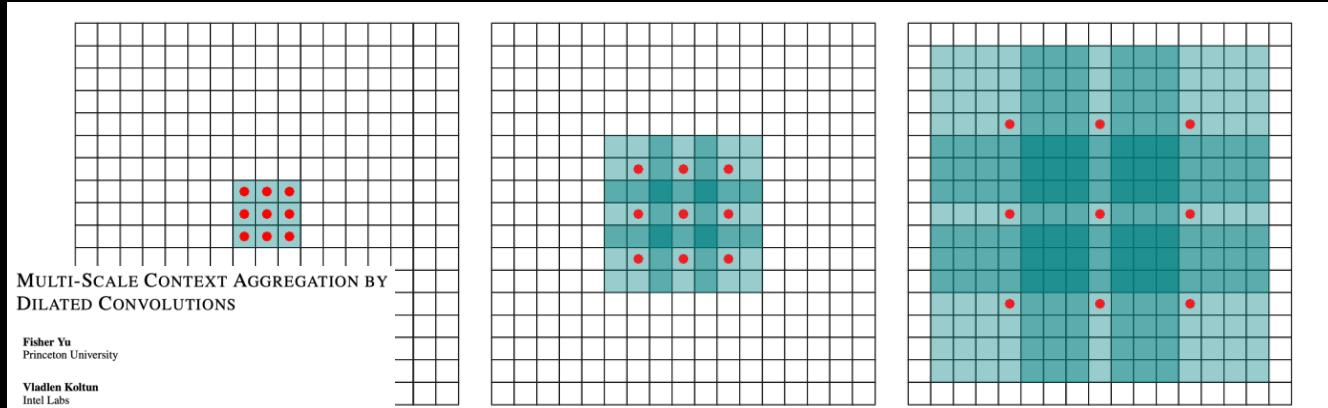
Pyramid Scene Parsing Network

Hengshuang Zhao¹ Jianping Shi² Xiaojian Qi¹ Xiaogang Wang¹ Jiaya Jia¹
¹The Chinese University of Hong Kong ²SenseTime Group Limited
{hszhao, xjqi, leoja}@cse.cuhk.edu.hk, xgwang@ee.cuhk.edu.hk, shjianping@sensetime.com

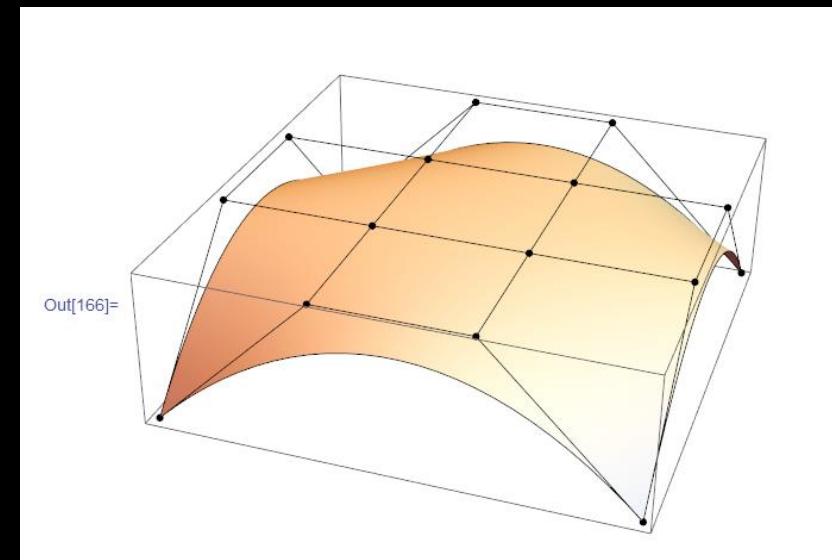
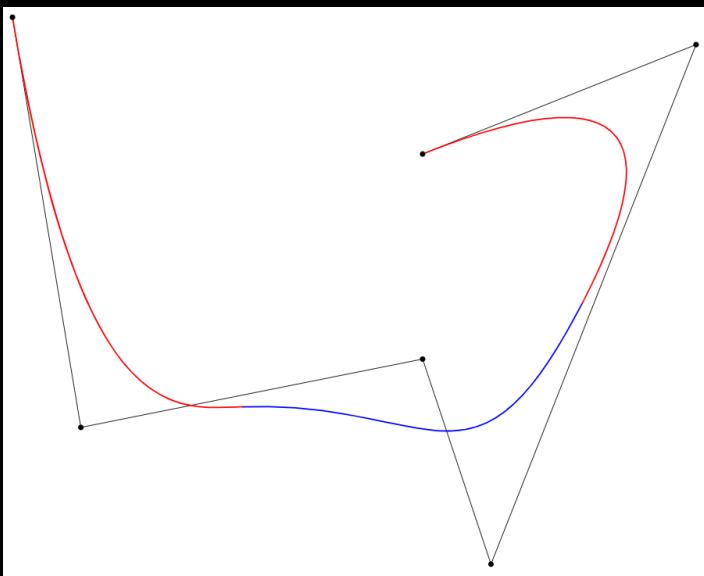
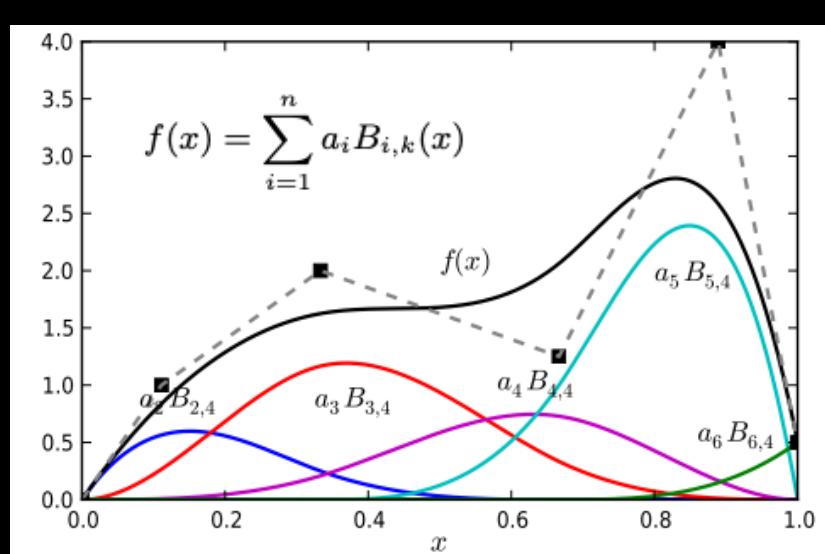


Semantic Segmentation: lessons learned

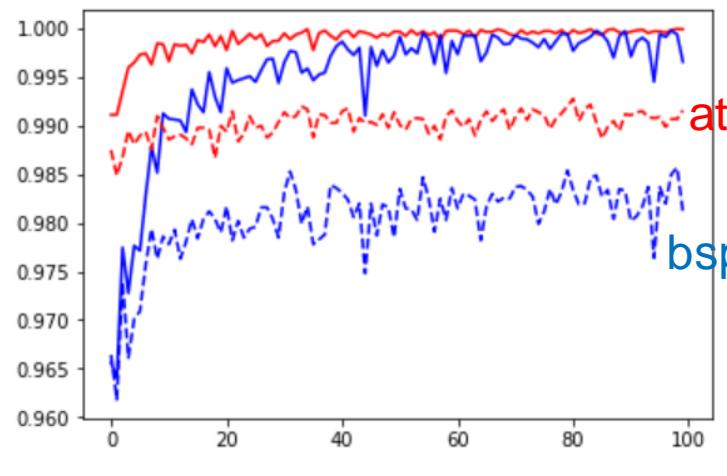
atrous convolutions - increased receptive field and/or regularization?



Bspline convolution: an interesting idea that failed (but we think we learned something 😊 - not published)



Semantic Segmentation: lessons learned



```

class net1(Block):
    def __init__(self,**kwargs):
        Block.__init__(self,**kwargs)
        with self.name_scope():

            self.conv1 = gluon.nn.Conv2D(channels=20, kernel_size=3, dilation=(2,2), activation='relu')
            self.pool1 = gluon.nn.MaxPool2D(pool_size=2, strides=2)
            self.BN1 = gluon.nn.BatchNorm()

            self.conv2 = gluon.nn.Conv2D(channels=50, kernel_size=3, dilation=(2,2), activation='relu')
            self.pool2 = gluon.nn.MaxPool2D(pool_size=2, strides=2)
            self.BN2 = gluon.nn.BatchNorm()

            self.flat = gluon.nn.Flatten()
            self.dense1 = gluon.nn.Dense(num_fc, activation="relu")
            self.dense2 = gluon.nn.Dense(num_outputs)

    def forward(self,_x):
        x1 = self.conv1(_x)
        x1 = self.BN1(x1)
        x = self.pool1(x1)

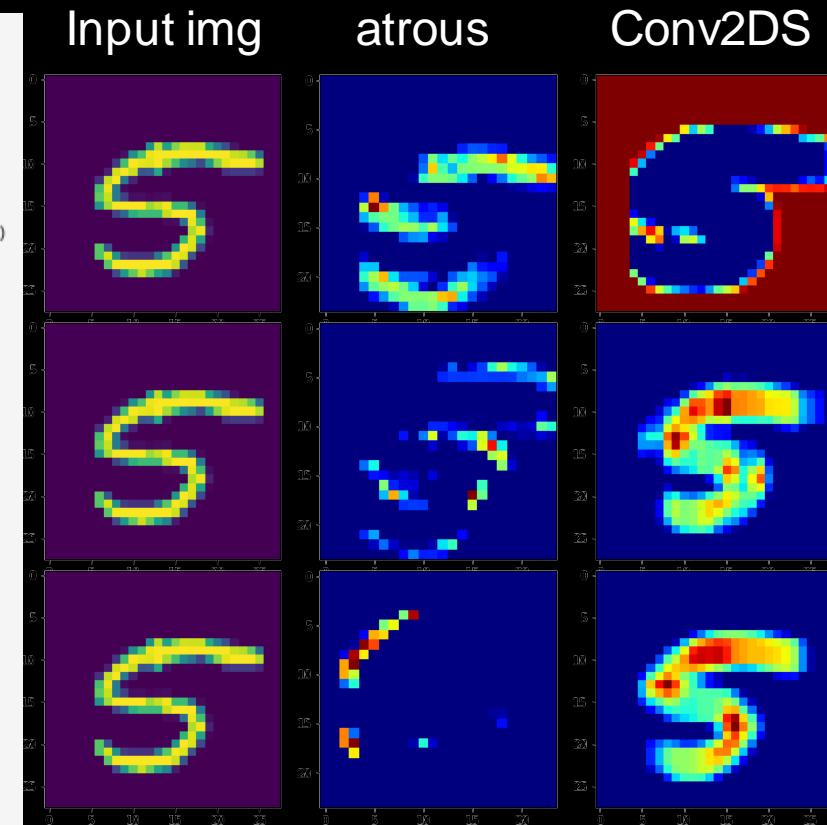
        x2 = self.conv2(x)
        x2 = self.BN2(x2)
        x2 = nd.relu(x2)
        x = self.pool2(x2)

        x = self.flat(x)
        x = self.dense1(x)
        x = self.dense2(x)

        return x, x1, x2

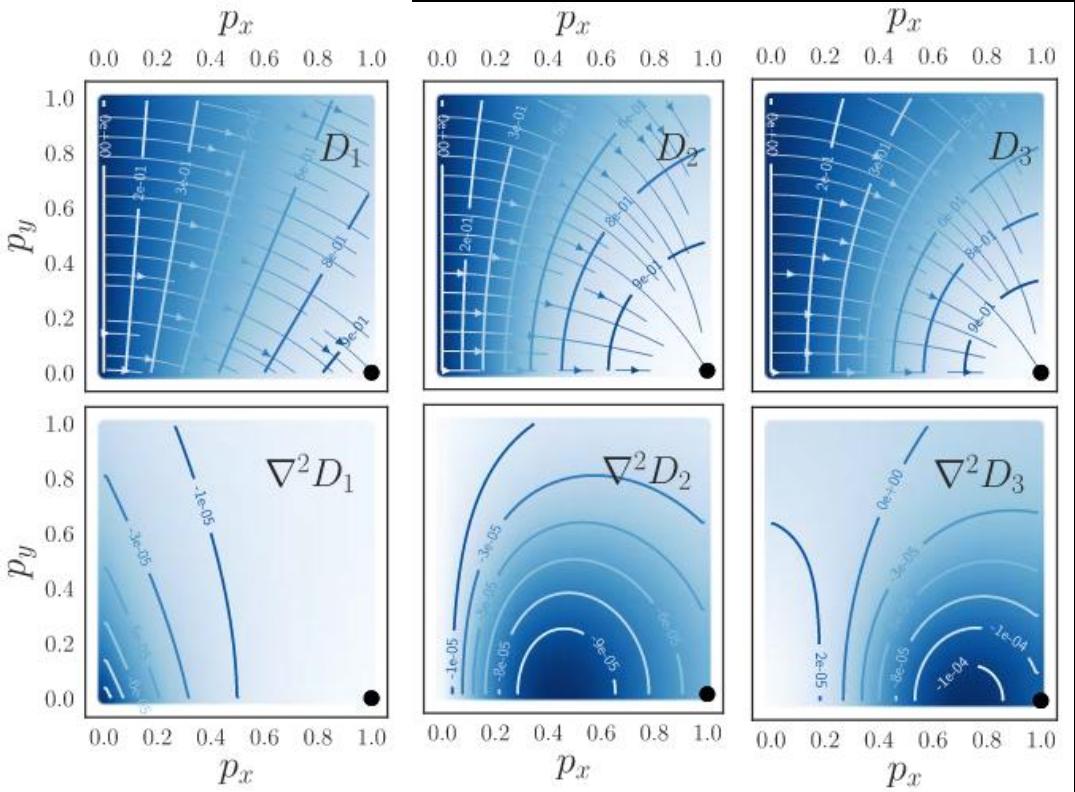
```

atrous convolutions - increased receptive field and/or regularization?



Semantic Segmentation: lessons learned – Dice, a set similarity metric

Not all “Dice” are created equal ...



$$D_1(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i p_i + \sum_i l_i}$$

$$D_2(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2)}$$

$$D_3(\mathbf{p}, \mathbf{l}) = \frac{\sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2) - \sum_i (p_i l_i)}$$

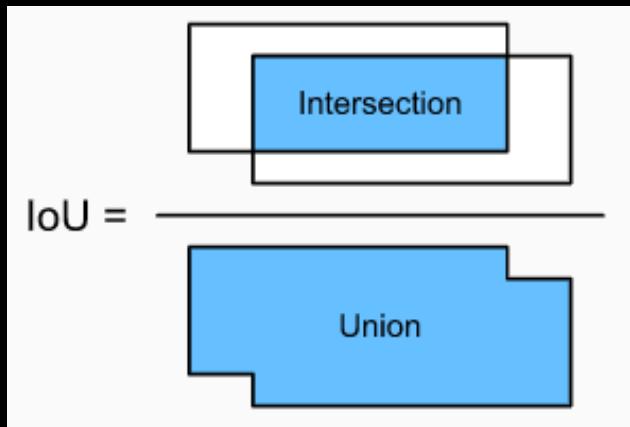


Image credit: d2l.ai

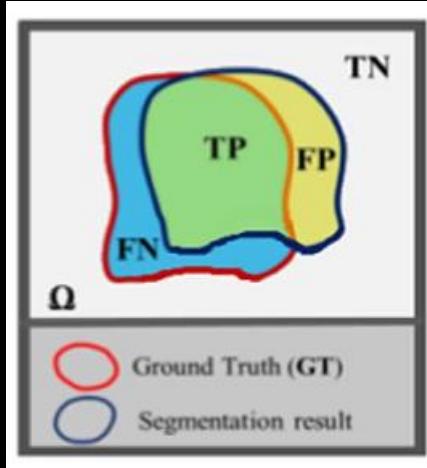
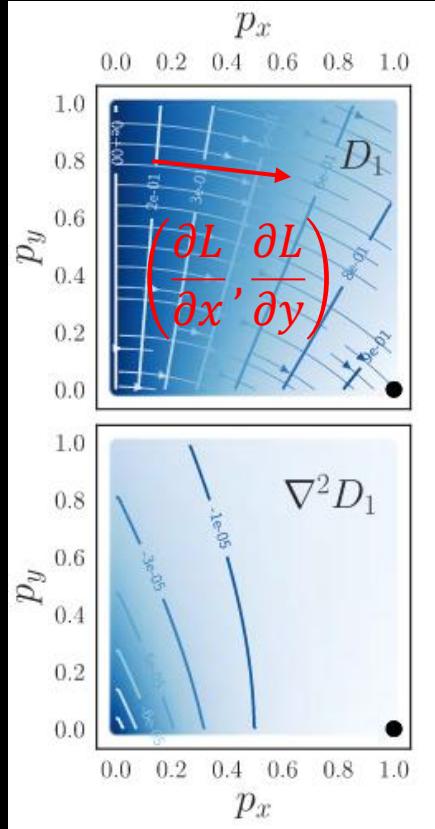
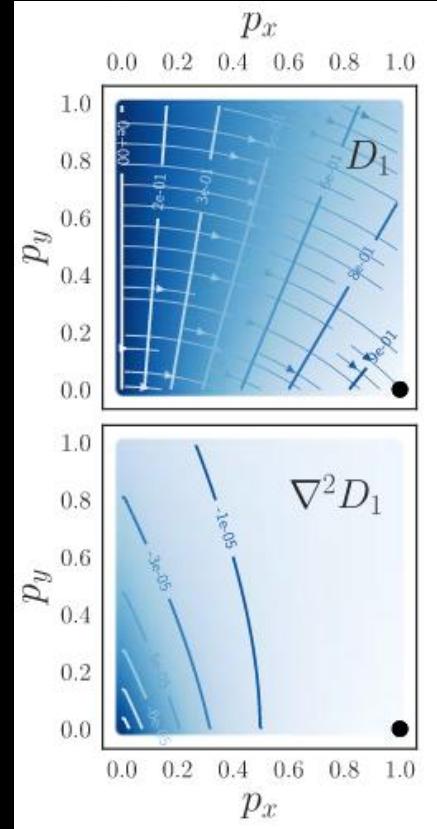


Image credit, Islam Reda et al. 2017,
A comprehensive non-invasive framework for diagnosing prostate cancer



Loss derivatives w.r.t. last layer:
a tangent vector

hidden layers

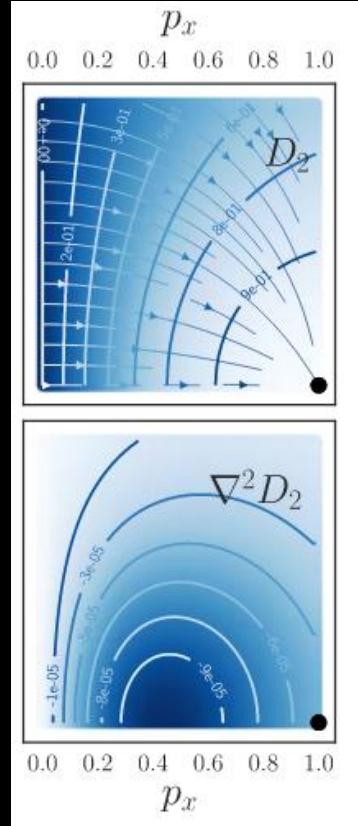
Inner product

$$\frac{\partial L}{\partial \vec{w}} = \dots \overbrace{\left(\frac{\partial x}{\partial \vec{w}}, \frac{\partial y}{\partial \vec{w}} \right)}^{\text{hidden layers}} \circ \underbrace{\left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y} \right)}_{\text{last layer}}$$

$$D_1(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i p_i + \sum_i l_i}$$

$$D_2(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2)}$$

$$D_3(\mathbf{p}, \mathbf{l}) = \frac{\sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2) - \sum_i (p_i l_i)}$$

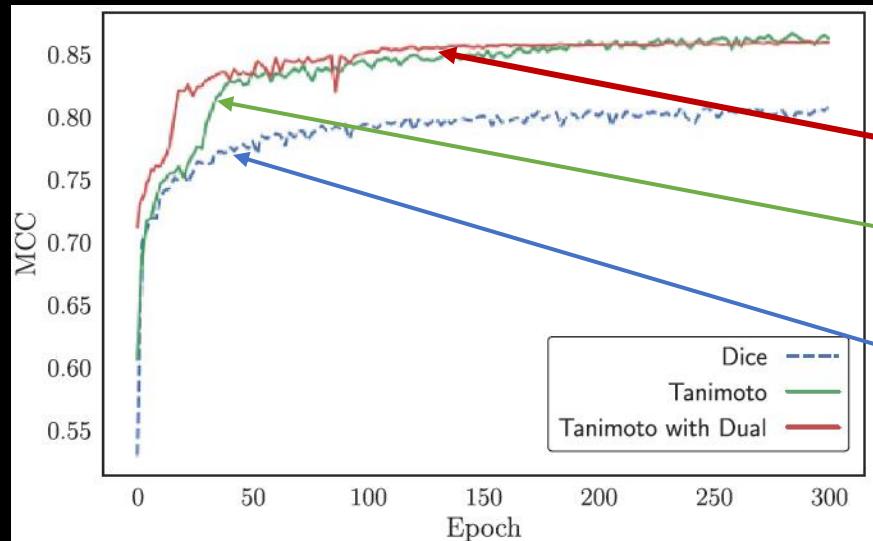


Not all “Dice” are created equal ...

Semantic Segmentation: lessons learned – Tanimoto set similarity metric

$$\tilde{T}(p_i, l_i) = \frac{T(p_i, l_i) + T(1 - p_i, 1 - l_i)}{2}$$

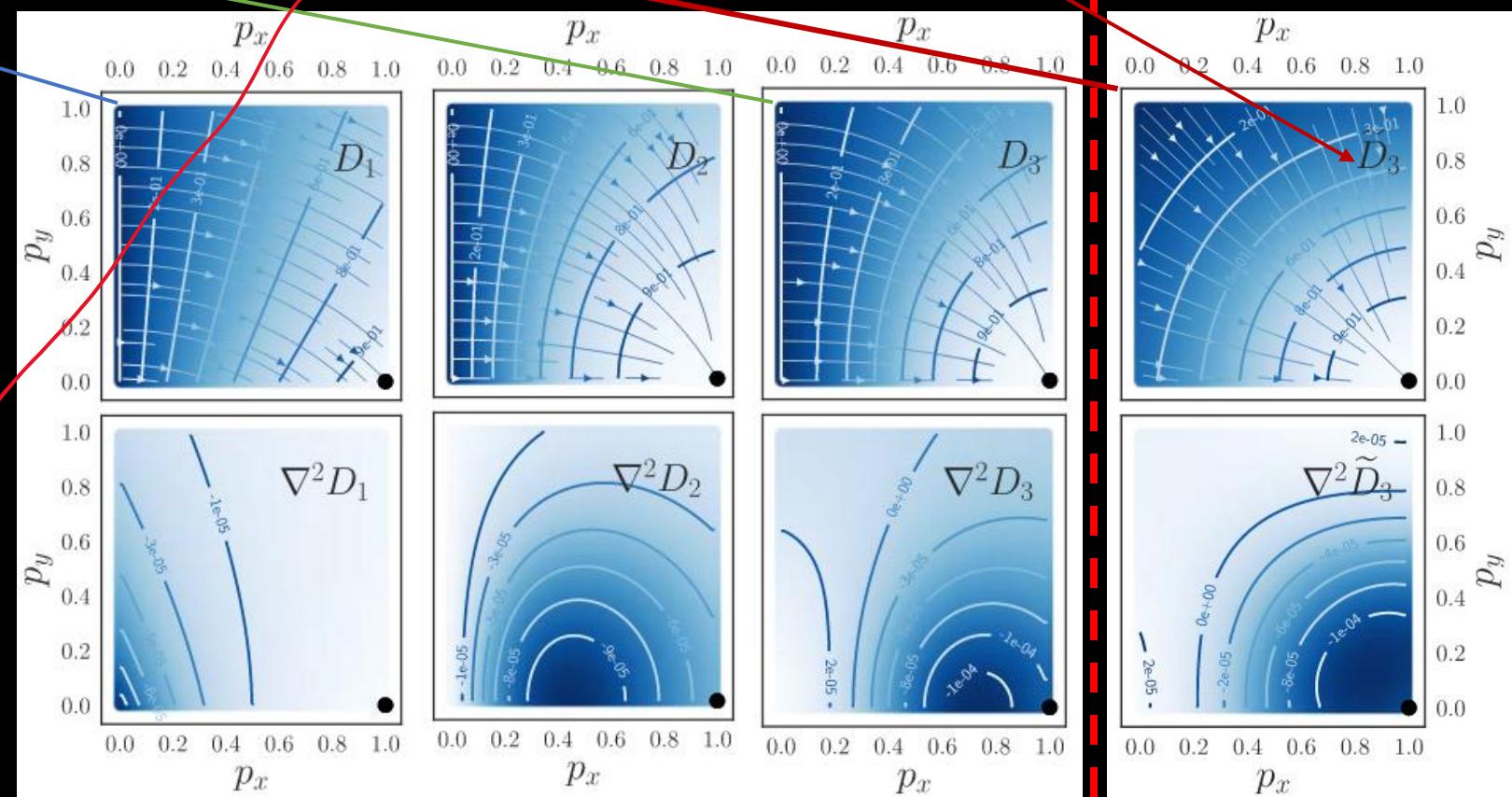
ResUNet - a Diakogiannis et al. 2019



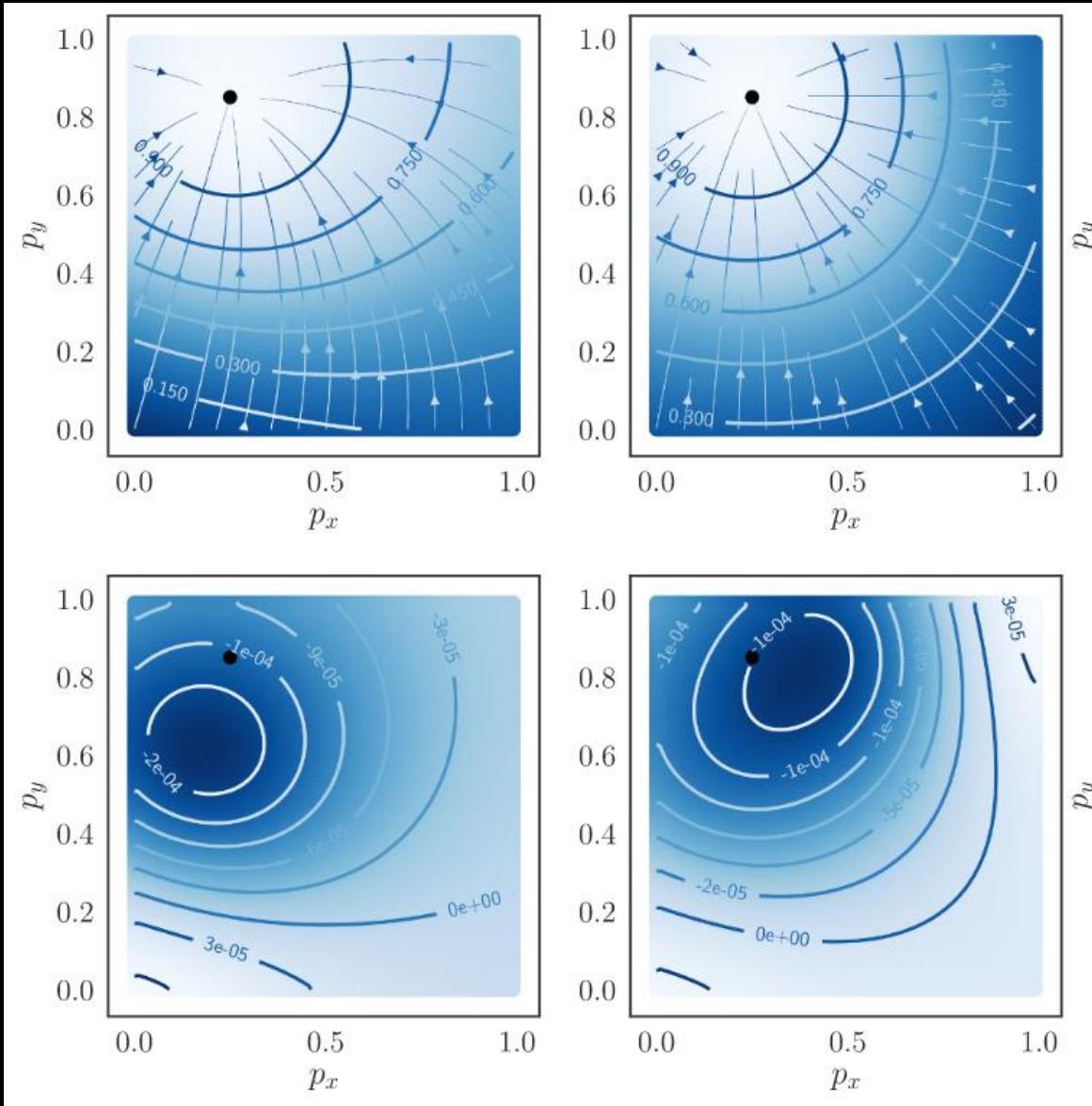
$$D_1(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i p_i + \sum_i l_i}$$

$$D_2(\mathbf{p}, \mathbf{l}) = \frac{2 \sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2)}$$

$$D_3(\mathbf{p}, \mathbf{l}) = \frac{\sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2) - \sum_i (p_i l_i)}$$

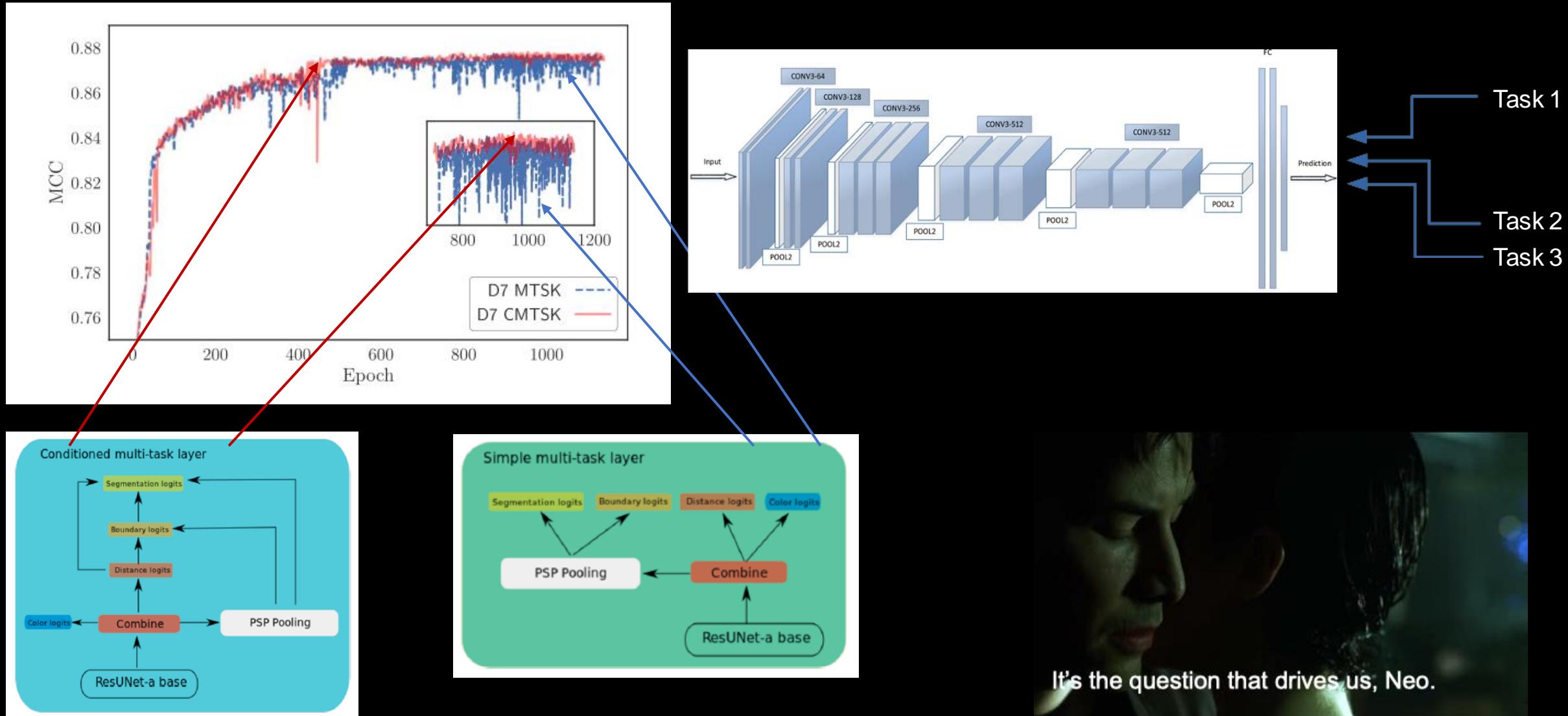


Semantic Segmentation: lessons learned – Tanimoto set similarity metric

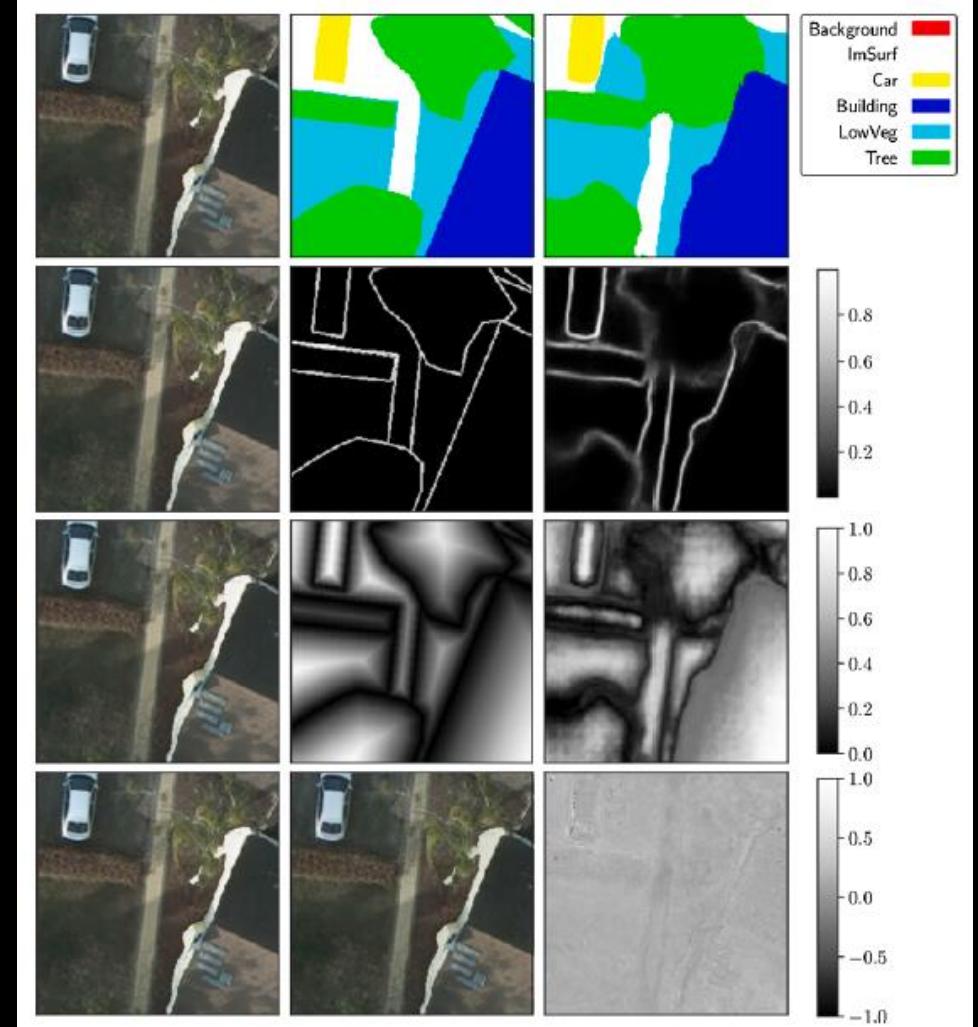
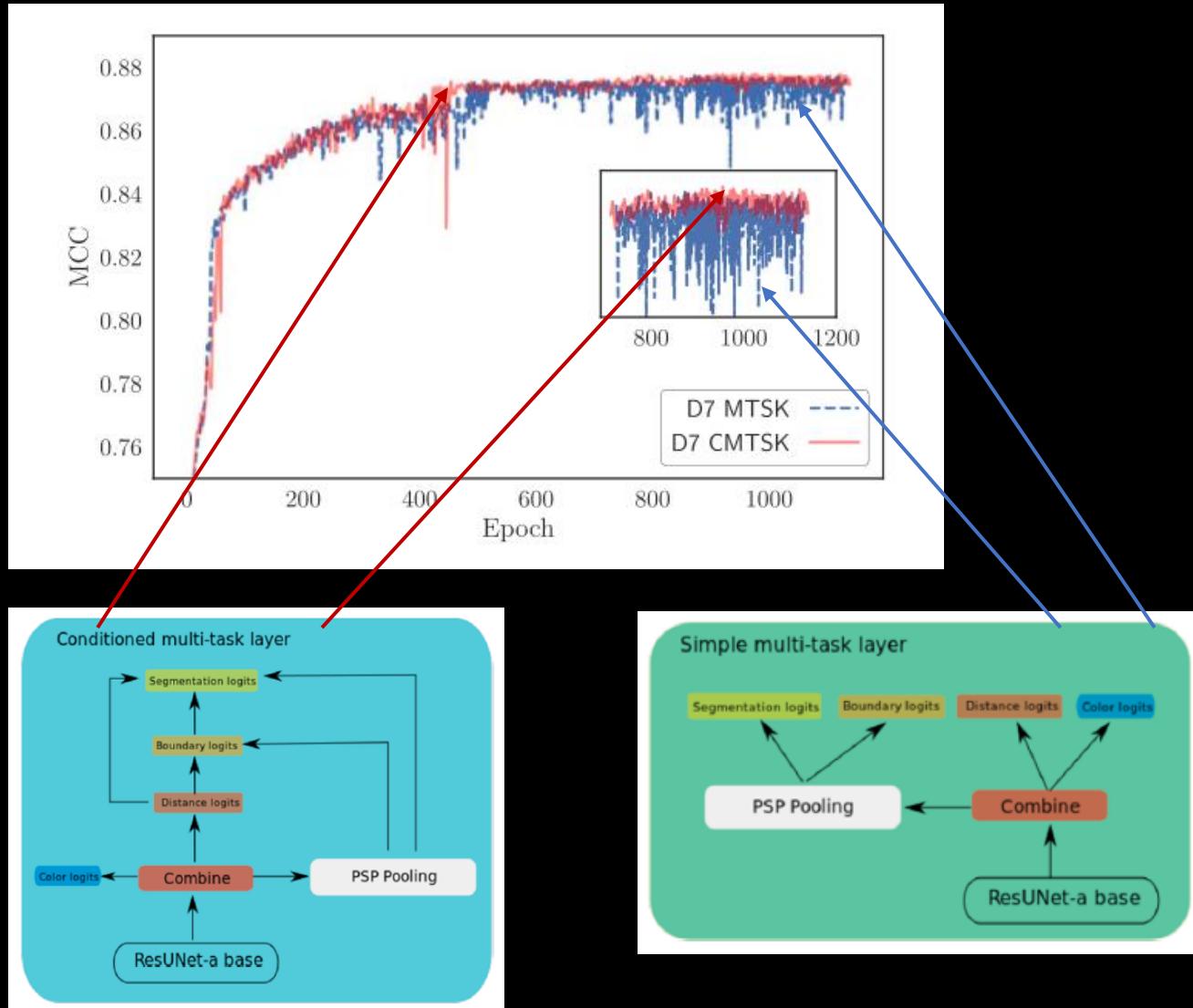


Still, not spot on, not steep enough ...
Can we improve?

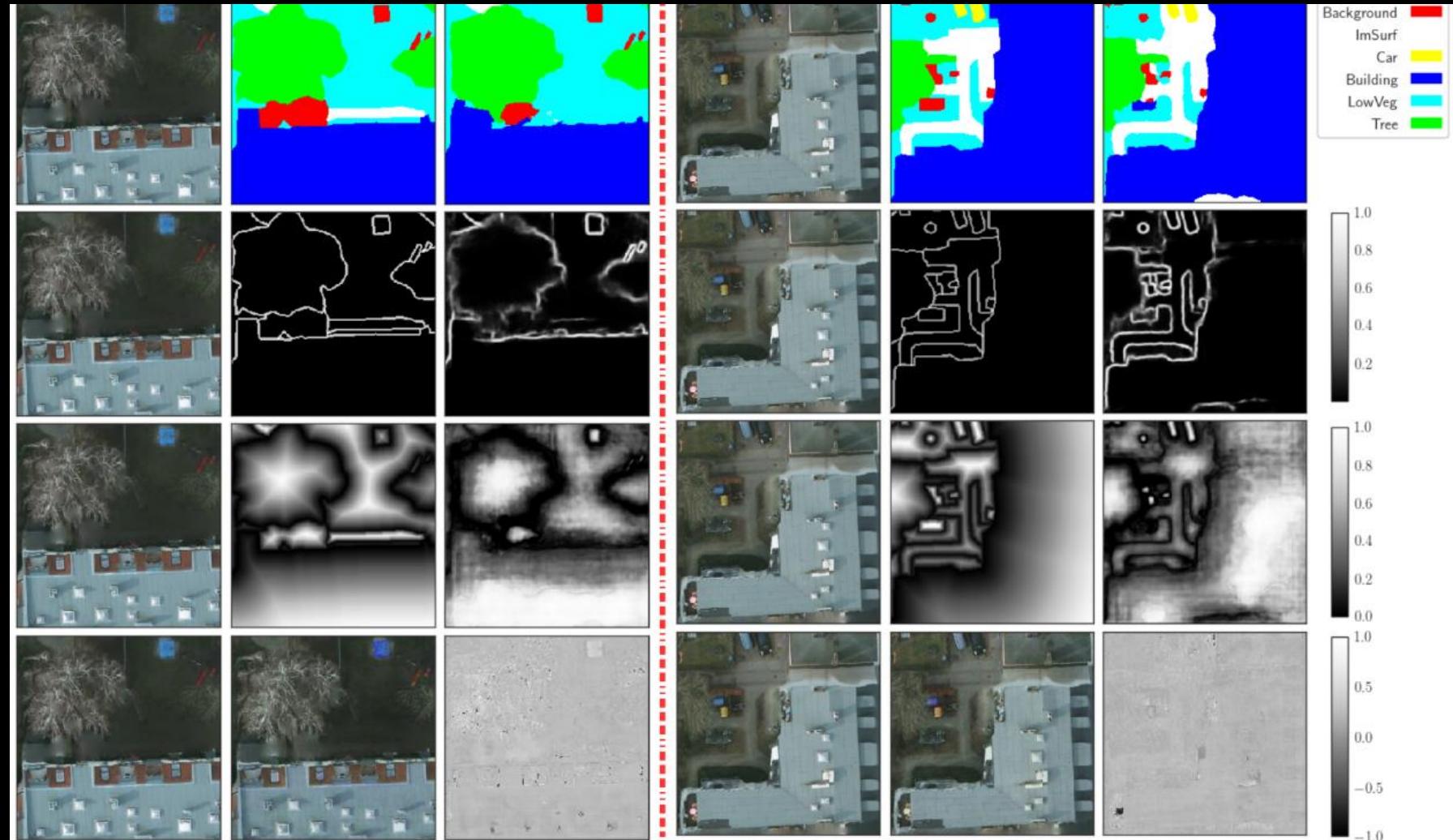
Semantic Segmentation: lessons learned – Multitasking, “causal” vs non-“causal”



Semantic Segmentation: lessons learned - “causal” multitasking semantic segmentation

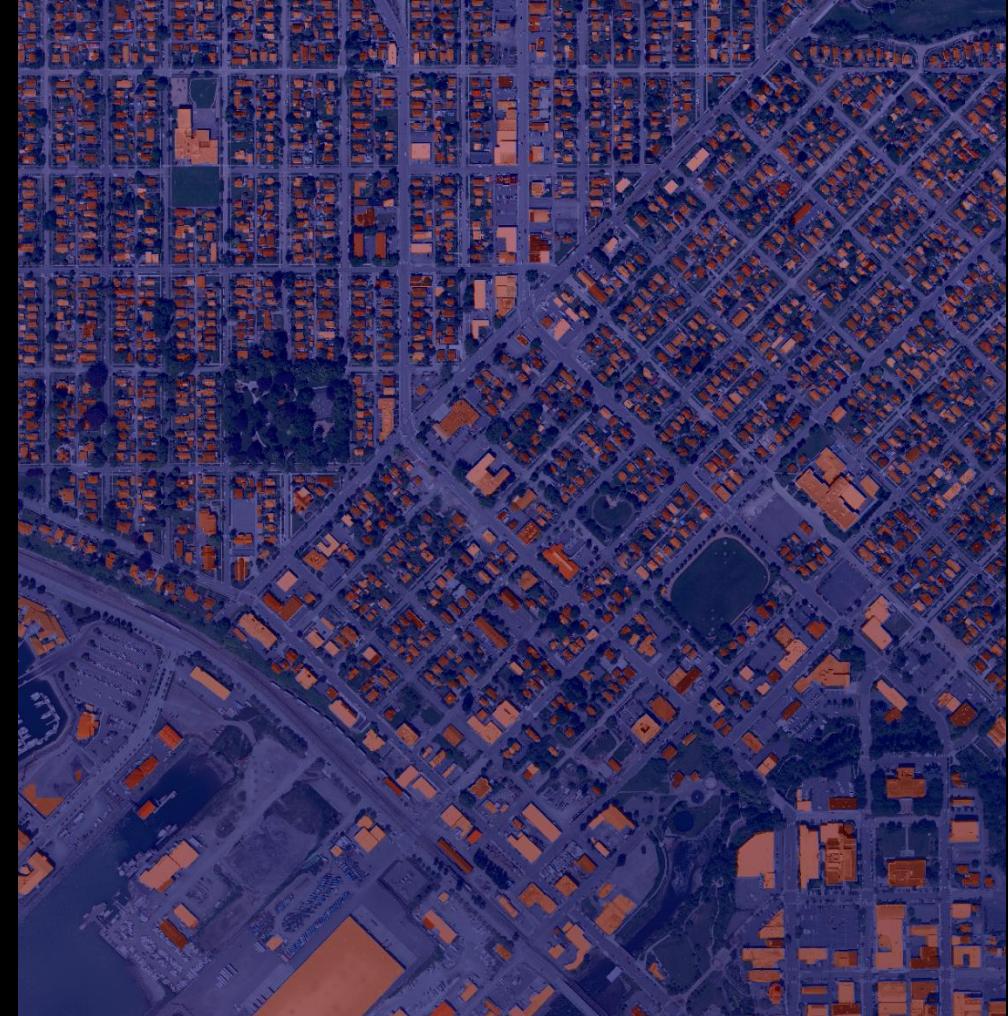


Multitasking: Questions guide performance



Semantic Segmentation: lessons learned – emphasis on boundaries – need sharp transition

Can we improve? (Inria building dataset, early attempts, unpublished)



Change detection: **double** input **semantic segmentation**



Looking for change? Roll the Dice and demand Attention

Foivos I. Diakogiannis^{a,b,1}, François Waldner^c, Peter Caccetta^b

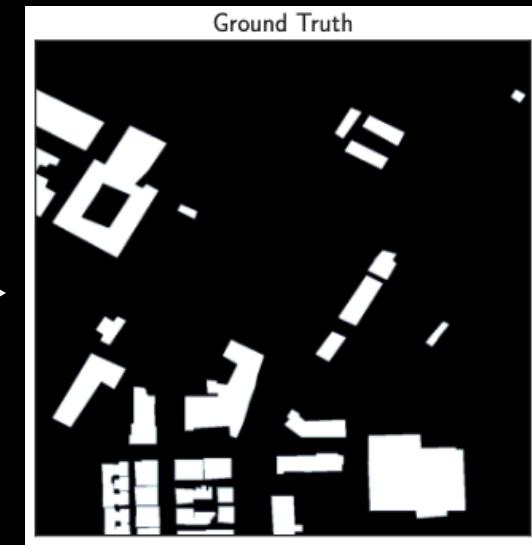
^aICRAR, the University of Western Australia

^bData61, CSIRO, Floreat WA

^cCSIRO Agriculture & Food, St Lucia, QLD, Australia

under review

Compare (DL)



Putting all things together, how do we:

- combine/process/consume/compare the two inputs?
- Improve on the segmentation task?

Change detection: learning to compare



Identify change between two bi-temporal images, on objects of interest.



More complex images take more time to identify where change exists

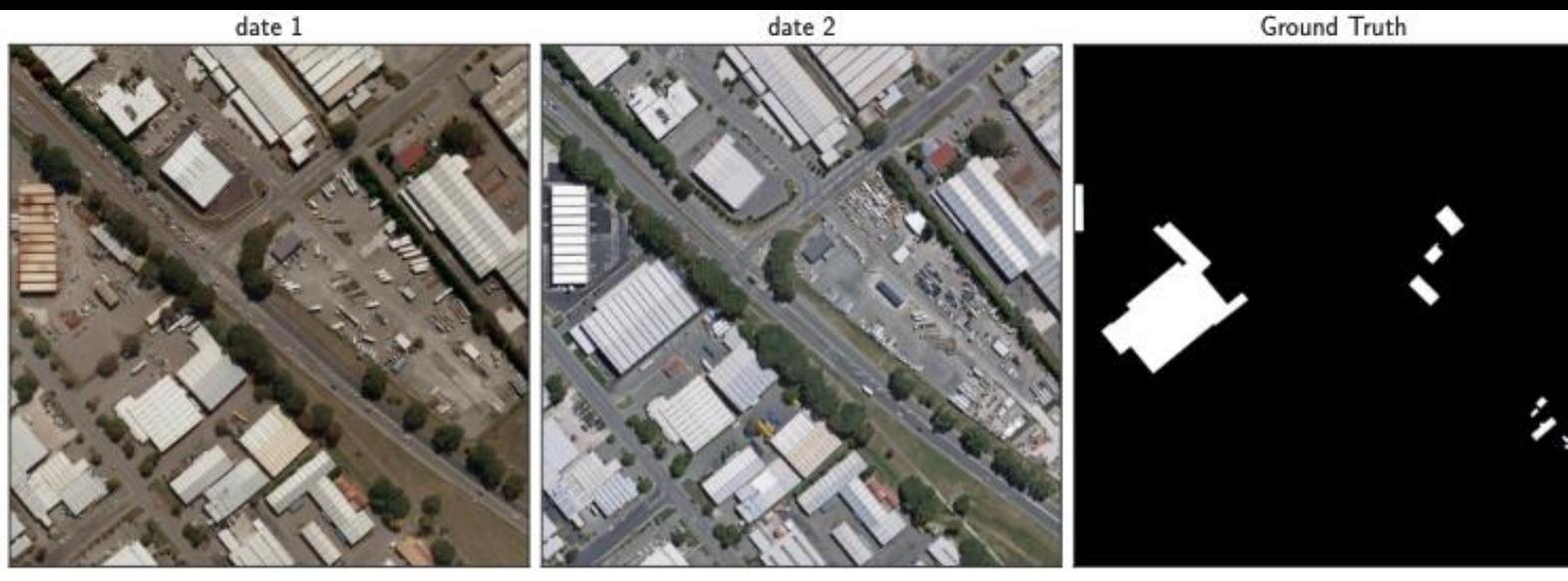
Change detection: learning to compare



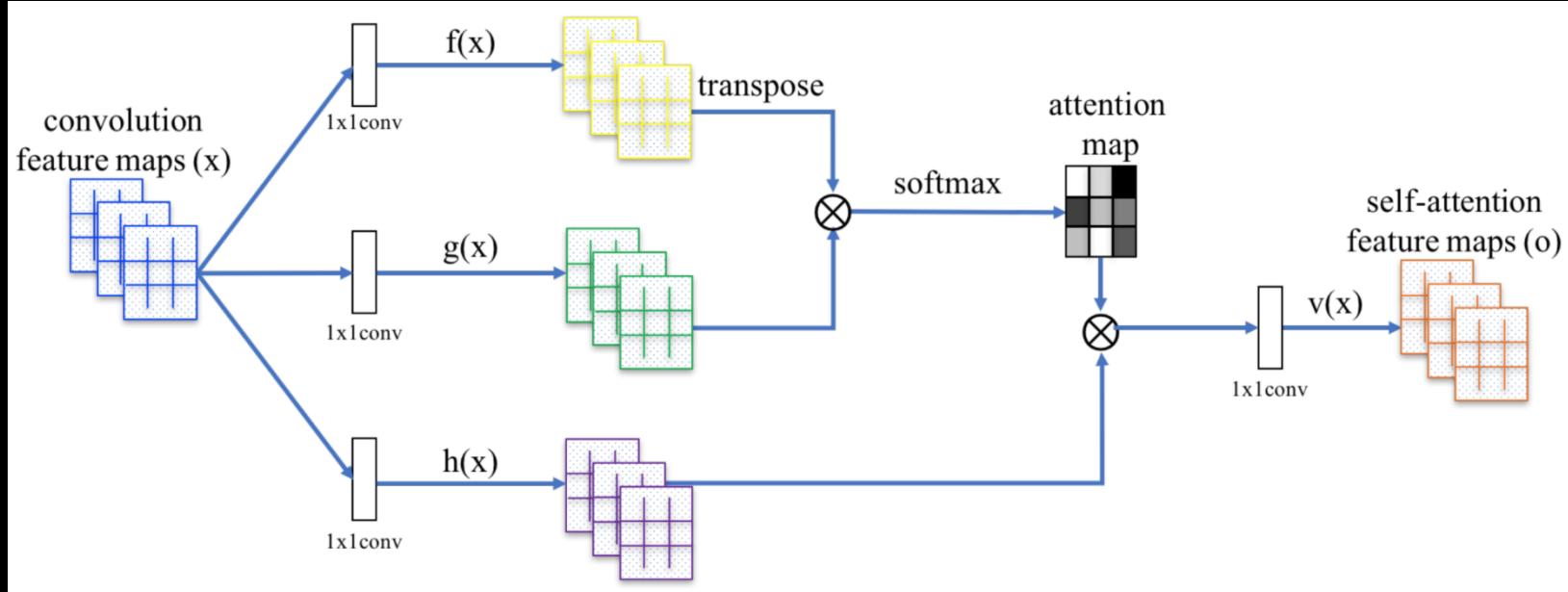
- Hypothesis: identifying change is a higher cognitive process.
- We understand this because time to compare correlates strongly with the complexity of change.
- Change cannot be identified by subtraction (constant time operation → contradicts complexity of time correlation).
- Hypothesis: we need a mechanism similar to **Attention**.



Change detection: learning to compare



On self Attention from NLP + vision (SAGAN)



Attention (NLP) Vaswani et al. (2017)

$$\mathbf{o} = \text{softmax}\left(\frac{\mathbf{q} \circ_1 \mathbf{k}}{\sqrt{d}}\right), \quad \in \Re^{C_q \times C}$$

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathbf{o} \circ_2 \mathbf{v}, \quad \in \Re^{C_q \times H \times W}$$

$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} = \mathbf{Z}$$

Quest:

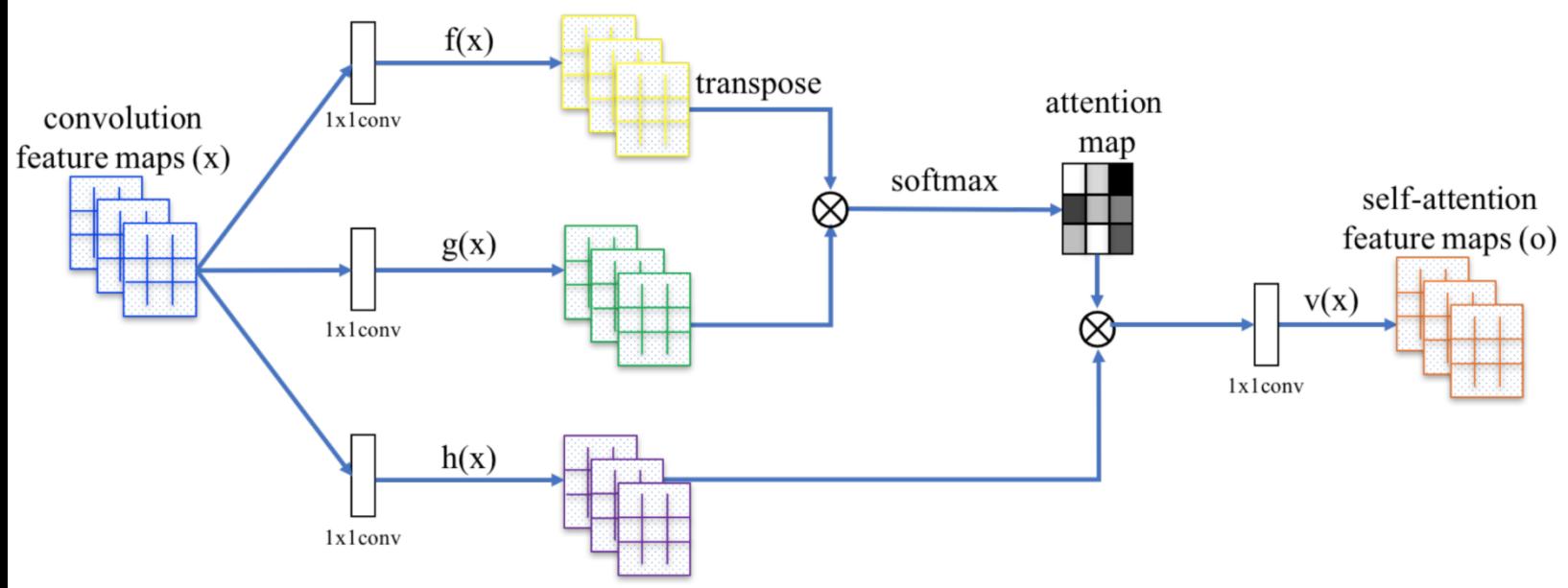
1. Understand Attention... (improve on atrous + use it for comparison)
2. Dot product is not a nice comparison operator (not bounded, not set similarirt)
3. Attention (memory) is about set operations:
need LAYER set similarity
4. Memory efficient Attention

$$\mathbf{q} \circ_1 \mathbf{k} \equiv \sum_{jk} q_{ijk} k_{ljk} \in \Re^{C_q \times C}$$

$$\mathbf{o} \equiv \text{softmax}(\mathbf{q} \circ_1 \mathbf{k}) \equiv o_{il}$$

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) \equiv \text{Att}_{ikj} \equiv \mathbf{o} \circ_2 \mathbf{v} \equiv \sum_l o_{il} v_{lkj} \in \Re^{C_q \times H \times W}$$

On self Attention from NLP + vision (SAGAN)



Attention (NLP) Vaswani et al. (2017)

$$\mathbf{o} = \text{softmax}\left(\frac{\mathbf{q} \circ_1 \mathbf{k}}{\sqrt{d}}\right), \quad \in \Re^{C_q \times C}$$

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathbf{o} \circ_2 \mathbf{v}, \quad \in \Re^{C_q \times H \times W}$$

$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V} = \mathbf{Z}$$

Quest:

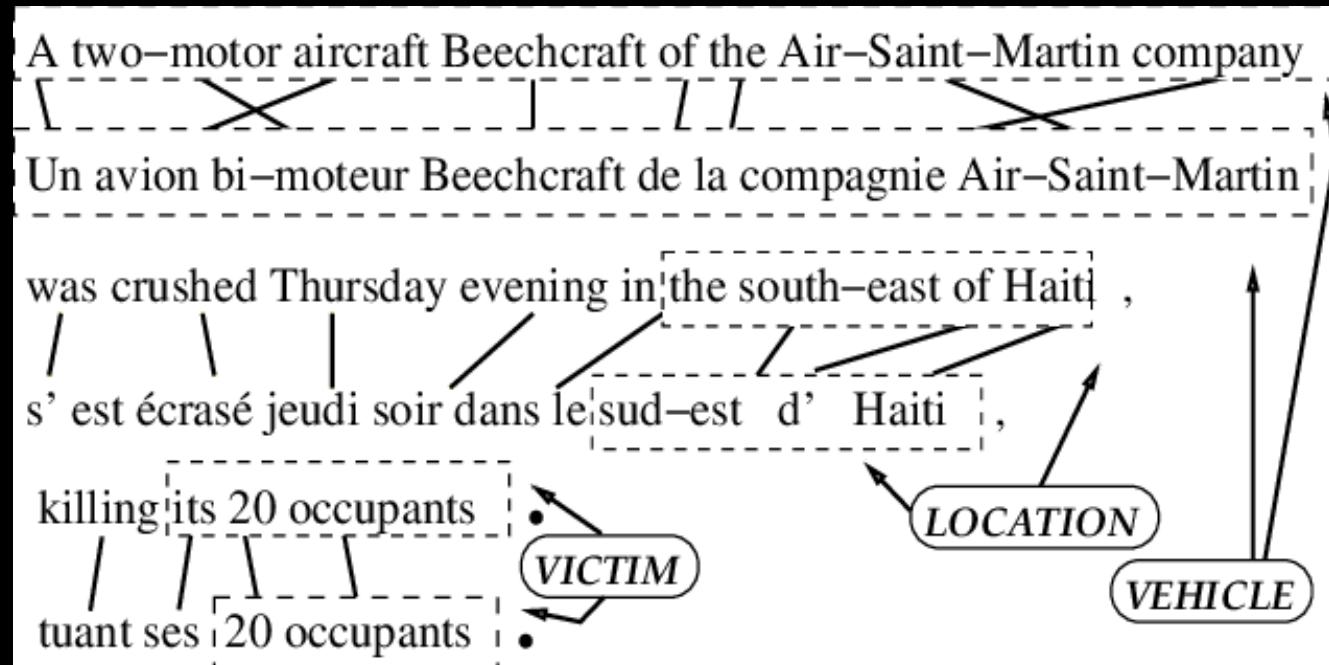
1. Understand Attention... (improve on atrous + use it for comparison)
2. Dot product is not a nice comparison operator (not bounded, not set similarity)
3. Attention (memory) is about set operations:
need LAYER set similarity
4. Memory efficient Attention ← dot product

Large number of filters and/or large spatial dimensions → GPU memory problem e.g. ResUNet 6th depth layer (input: 4 x 32 x 256 x 256):

\mathbf{Q} (4 x 1024 x 8 x 8)
 \mathbf{K} (4 x 1024 x 8 x 8)

$\text{Dot}(\mathbf{Q}, \mathbf{K}) \sim 4 \times 1024 \times 1024$

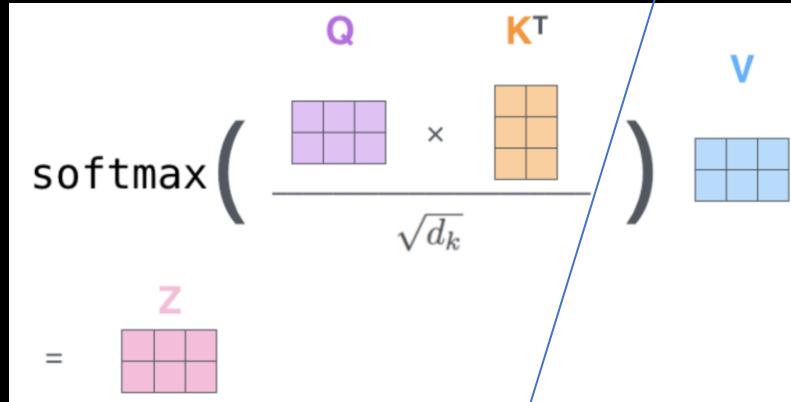
On self Attention from NLP + vision (SAGAN)



Attention (NLP) Vaswani et al. (2017)

$$\mathbf{o} = \text{softmax}\left(\frac{\mathbf{q} \circ_1 \mathbf{k}}{\sqrt{d}}\right), \quad \in \mathbb{R}^{C_q \times C}$$

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathbf{o} \circ_2 \mathbf{v}, \quad \in \mathbb{R}^{C_q \times H \times W}$$



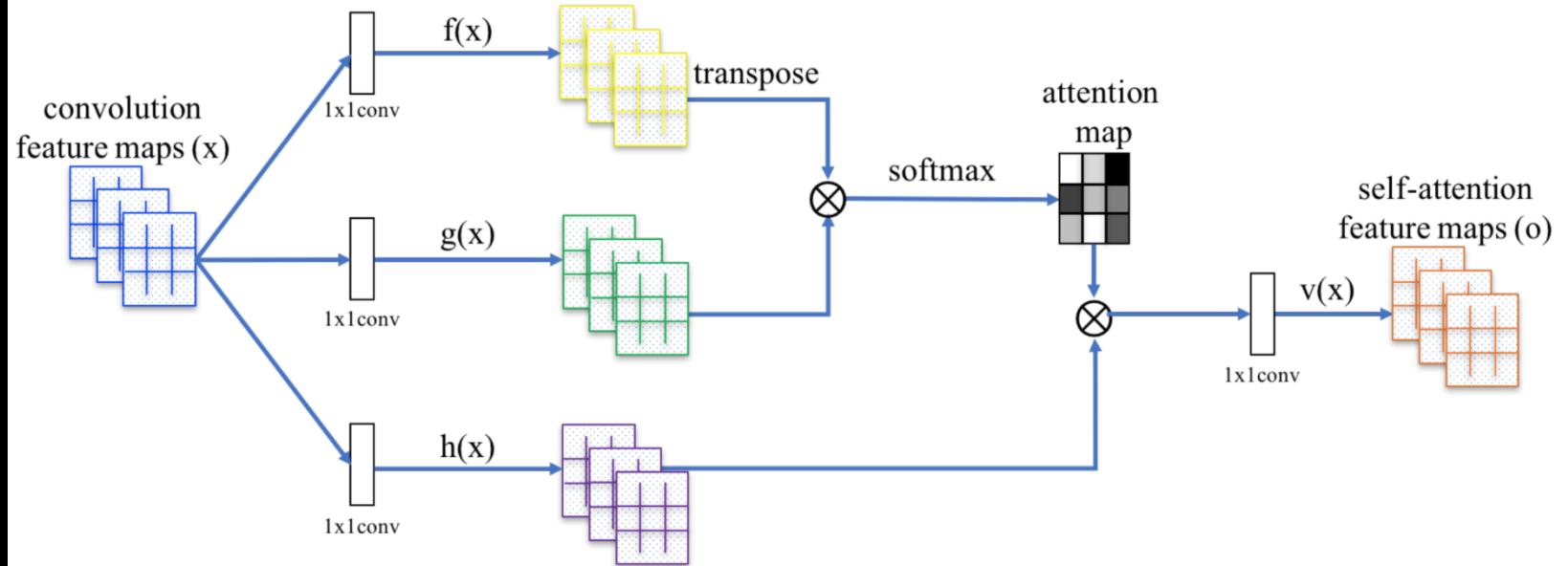
Quest:

1. Understand Attention... (improve on atrous + use it for comparison)
2. Dot product is not a nice comparison operator (not bounded, not set similarity)
3. Attention (memory) is about set operations:
need LAYER set similarity
4. Memory efficient Attention ↘ dot product

In NMT relative position is important (syntax changes). In vision this is not necessarily the case: element wise channel comparison → memory efficiency

On (traditional) Attention (from NLP)

Attention (NLP) Vaswani et al. (2017)



$$\mathbf{o} = \text{softmax}\left(\frac{\mathbf{q} \circ_1 \mathbf{k}}{\sqrt{d}}\right), \quad \in \Re^{C_q \times C}$$

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathbf{o} \circ_2 \mathbf{v}, \quad \in \Re^{C_q \times H \times W}$$

$$\text{softmax}\left(\frac{\mathbf{Q} \times \mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V} = \mathbf{Z}$$

Quest:

1. Understand Attention.
2. Dot product is not a nice comparison operator (not bounded, not set similarity)
3. Attention (memory) is about set operations: need LAYER set similarity
4. Memory efficient Attention ← dot product

==> New Layer (set) similarity metric (replace dot)

==> New evolving Loss strategy

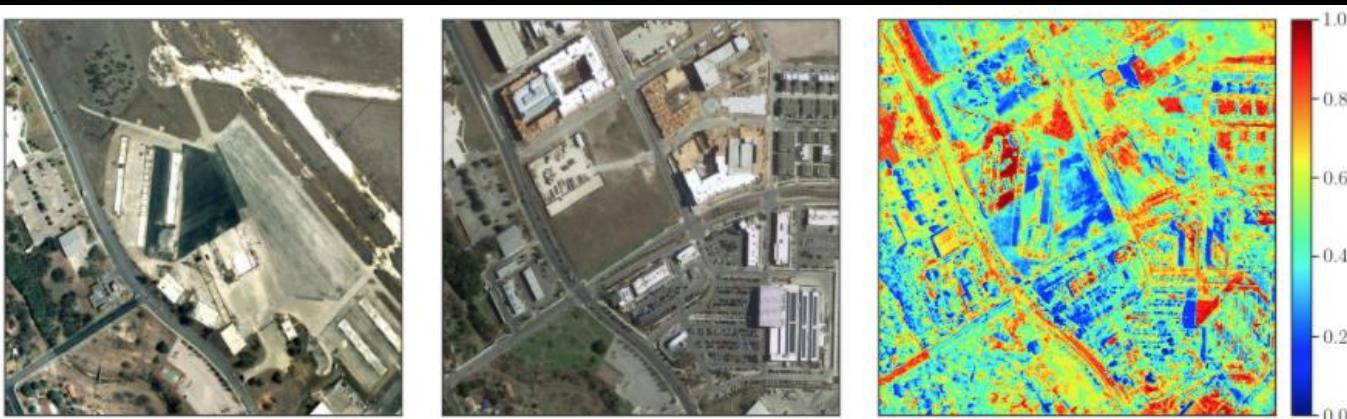
==> New Attention module (channel+spatial ==> memory efficient)

==> New attention-based building blocks (x2)

==> New treatment of boundaries

==> New change detection learning macro-topology

Dot product → Tanimoto set similarity (Diakogiannis et al. (2019))



A new set similarity: the FracTAL Tanimoto similarity

$$\mathcal{T}^0 \equiv T(\mathbf{p}, \mathbf{l}) = \frac{\mathbf{p} \cdot \mathbf{l}}{\mathbf{p}^2 + \mathbf{l}^2 - \mathbf{p} \cdot \mathbf{l}}$$

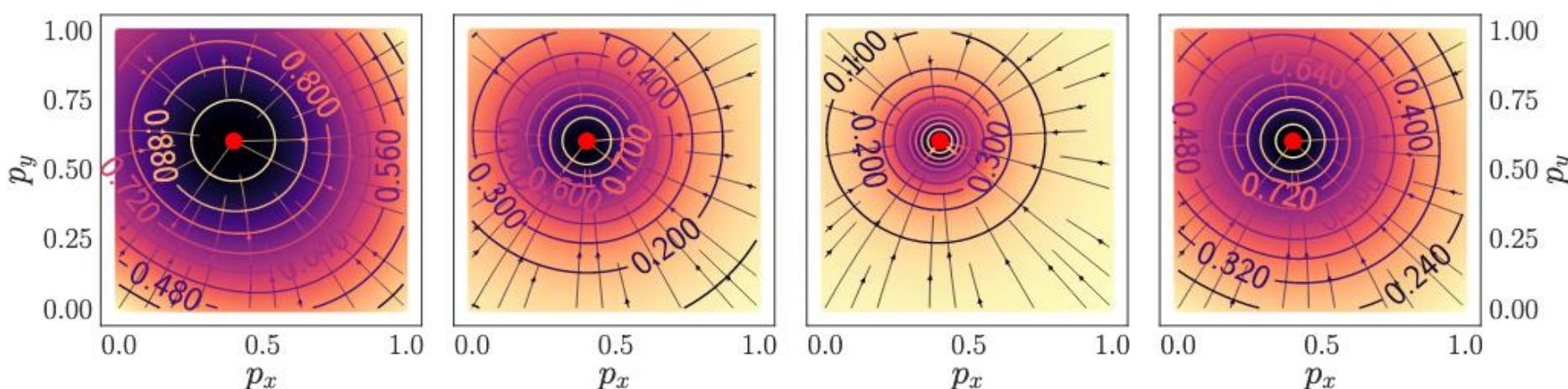
$$\mathcal{T}^d = \frac{\mathcal{T}^{d-1}(\mathbf{p}, \mathbf{l})}{\mathcal{T}^{d-1}(\mathbf{p}, \mathbf{p}) + \mathcal{T}^{d-1}(\mathbf{l}, \mathbf{l}) - \mathcal{T}^{d-1}(\mathbf{p}, \mathbf{l})}$$

$d=3$

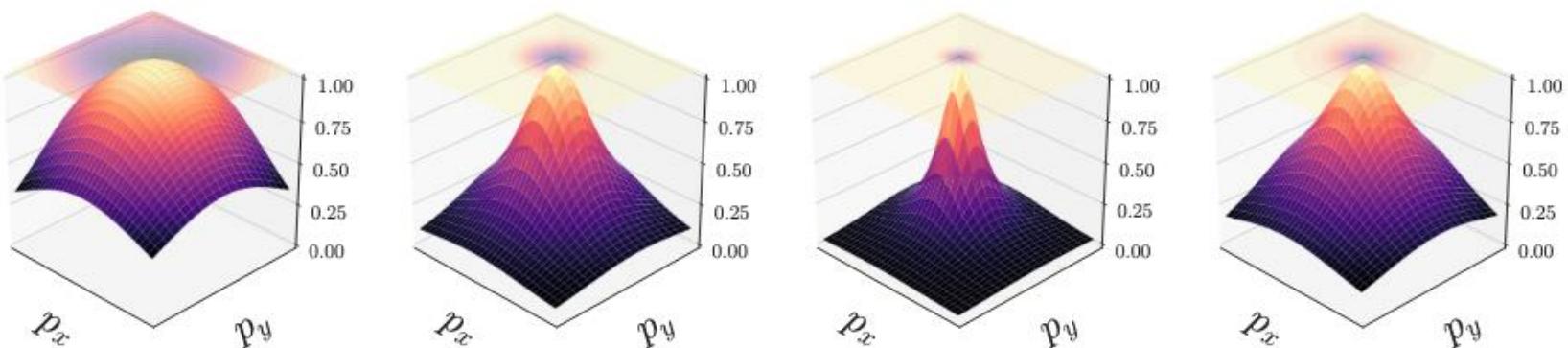


$$\mathcal{T}^3(\mathbf{p}, \mathbf{l}) = \frac{\mathbf{p} \cdot \mathbf{l}}{(\mathbf{l}^2 - \mathbf{p} \cdot \mathbf{l} + \mathbf{p}^2) \left(2 - \frac{\mathbf{p} \cdot \mathbf{l}}{\mathbf{l}^2 - \mathbf{p} \cdot \mathbf{l} + \mathbf{p}^2} \right) \left(2 - \frac{\mathbf{p} \cdot \mathbf{l}}{(\mathbf{l}^2 - \mathbf{p} \cdot \mathbf{l} + \mathbf{p}^2) \left(2 - \frac{\mathbf{p} \cdot \mathbf{l}}{\mathbf{l}^2 - \mathbf{p} \cdot \mathbf{l} + \mathbf{p}^2} \right)} \right)}$$

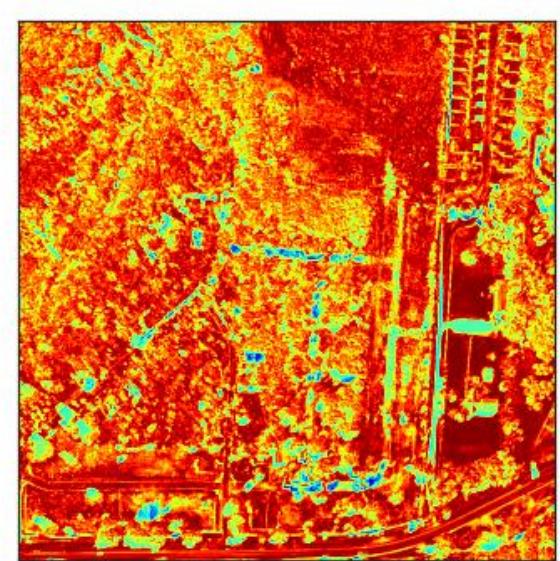
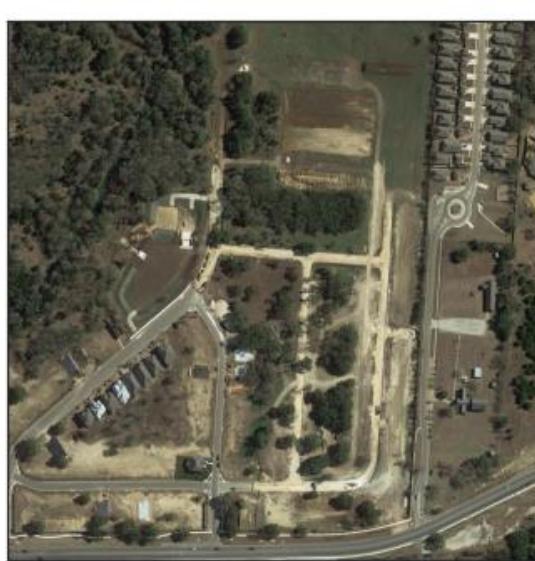
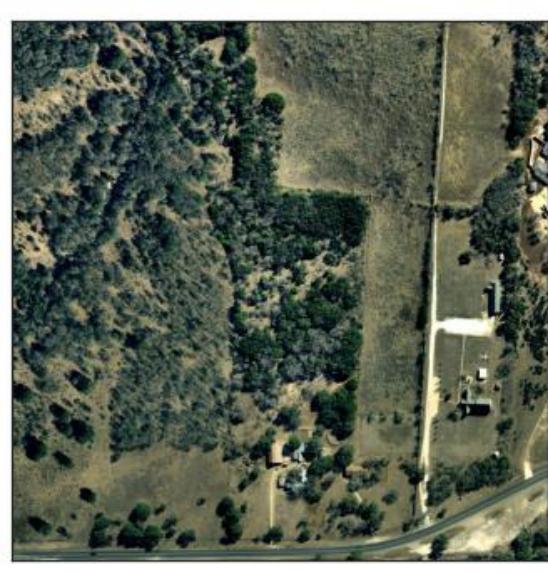
(4)



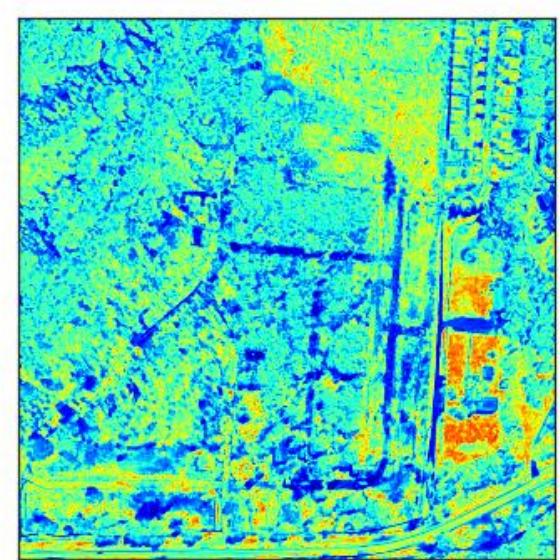
$$\mathcal{T}^d(\mathbf{p}, \mathbf{l}) = \frac{\mathbf{p} \cdot \mathbf{l}}{2^d(\mathbf{p}^2 + \mathbf{l}^2) - (2^{d+1} - 1)\mathbf{p} \cdot \mathbf{l}}$$



Finer layer (set) similarity (or else, why we did all this)



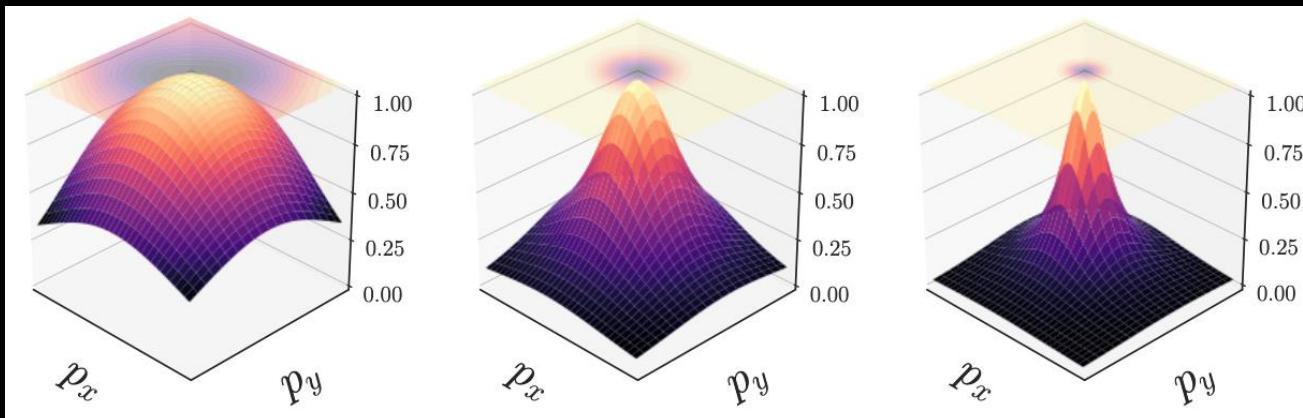
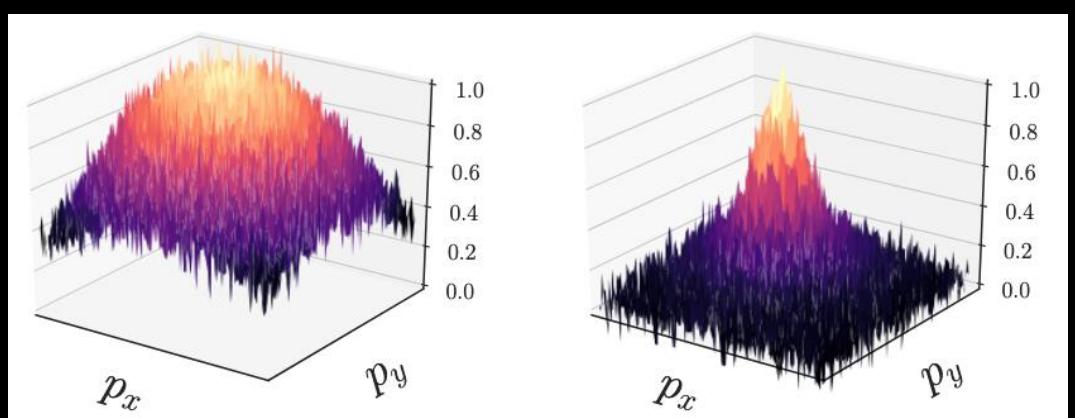
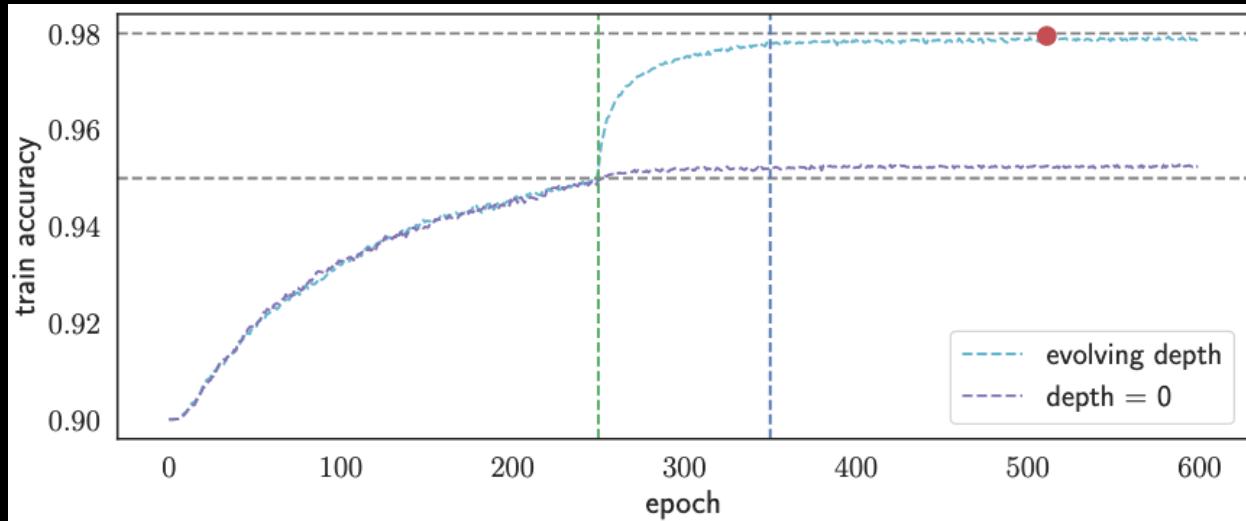
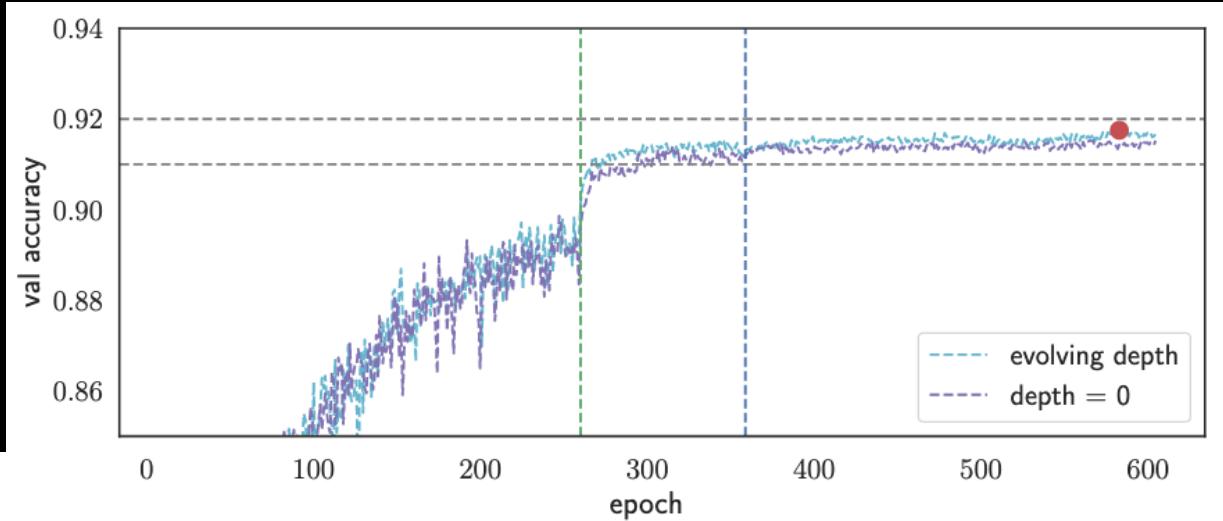
fractal depth=0



fractal depth=10

Corollary: from set similarity to evolving loss strategy

$$\mathcal{T}^d(\mathbf{p}, \mathbf{l}) = \frac{\mathbf{p} \cdot \mathbf{l}}{2^d(\mathbf{p}^2 + \mathbf{l}^2) - (2^{d+1} - 1)\mathbf{p} \cdot \mathbf{l}}$$



A new channel + spatial attention layer – the FracTAL

$$\text{Att}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = 0.5(\text{Att}_{\boxtimes} + \text{Att}_{\square})$$

$$\mathcal{T}_{\boxtimes}^d(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q} \boxtimes \mathbf{k}}{2^d (\mathbf{q} \boxtimes \mathbf{q} + \mathbf{k} \boxtimes \mathbf{k}) - (2^{d+1} - 1)\mathbf{q} \boxtimes \mathbf{k}} \in \Re^C \quad (11)$$

$$\mathcal{T}_{\square}^d(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q} \square \mathbf{k}}{2^d (\mathbf{q} \square \mathbf{q} + \mathbf{k} \square \mathbf{k}) - (2^{d+1} - 1)\mathbf{q} \square \mathbf{k}} \in \Re^{H \times W} \quad (12)$$



IoU – like similarity for binary images

```
class RelFTAttention2D(HybridBlock):
    def __init__(self, nkeys, kernel_size=3, padding=1, nheads=1, norm = 'BatchNorm', norm_groups=None, ftdepth=5, **kwargs):
        super().__init__(**kwargs)

        self.query = Conv2DNormed(channels=nkeys, kernel_size= kernel_size, padding = padding, _norm_type= norm, norm_groups=norm_groups)
        self.key = Conv2DNormed(channels=nkeys, kernel_size= kernel_size, padding = padding, _norm_type= norm, norm_groups=norm_groups)
        self.value = Conv2DNormed(channels=nkeys, kernel_size= kernel_size, padding = padding, _norm_type= norm, norm_groups=norm_groups)

        self.metric_channel = FTanimoto(depth=ftdepth, axis=[2,3])
        self.metric_space = FTanimoto(depth=ftdepth, axis=1)

        self.norm = get_norm(name=norm, axis=1, norm_groups= norm_groups)

    def forward(self, input1, input2, input3):

        # These should work with ReLU as well
        q = mx.npx.sigmoid(self.query(input1))
        k = mx.npx.sigmoid(self.key(input2))# B,C,H,W
        v = mx.npx.sigmoid(self.value(input3)) # B,C,H,W

        att_spat = self.metric_space(q,k) # B,1,H,W
        v_spat = att_spat * v # emphasize spatial features

        att_chan = self.metric_channel(q,k) # B,C,1,1
        v_chan = att_chan * v # emphasize spatial features

        v_cspat = 0.5*(v_chan + v_spat)
        v_cspat = self.norm(v_cspat)

        return v_cspat
```

$$\text{Att}_{\boxtimes}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathcal{T}_{\boxtimes}^d(\mathbf{q}, \mathbf{k}) \odot \mathbf{v}$$

$$\text{Att}_{\square}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \mathcal{T}_{\square}^d(\mathbf{q}, \mathbf{k}) \odot \mathbf{v}$$

```
from mxnet.gluon import nn
class FTanimoto(nn.Block):
    def __init__(self, depth=5, axis=[2,3], **kwargs):
        super().__init__(**kwargs)
        self.depth = depth
        self.axis=axis

    def inner_prod(self, prob, label):
        prdct = prob*label #dim:(B,C,H,W)
        prdct = prdct.sum(axis=self.axis,keepdims=True)
        return prdct #dim:(B,C,1,1)

    def forward(self, prob, label):
        a = 2.*self.depth
        b = -(2.*a-1.)

        tlp= self.inner_prod(prob,label)
        tpp= self.inner_prod(prob,prob)
        tll= self.inner_prod(label,label)

        denum = a*(tpp+tll)+b*tlp
        ftnmt = tlp/denum
        return ftnmt #dim:(B,C,1,1)
```

Self attention: how do you combine the FracTAL with a features layer?

```

def forward(self, input1, input2, input3):

    # These should work with ReLU as well
    q = mx.npx.sigmoid(self.query(input1))
    k = mx.npx.sigmoid(self.key(input2))# B,C,H,W
    v = mx.npx.sigmoid(self.value(input3)) # B,C,H,W

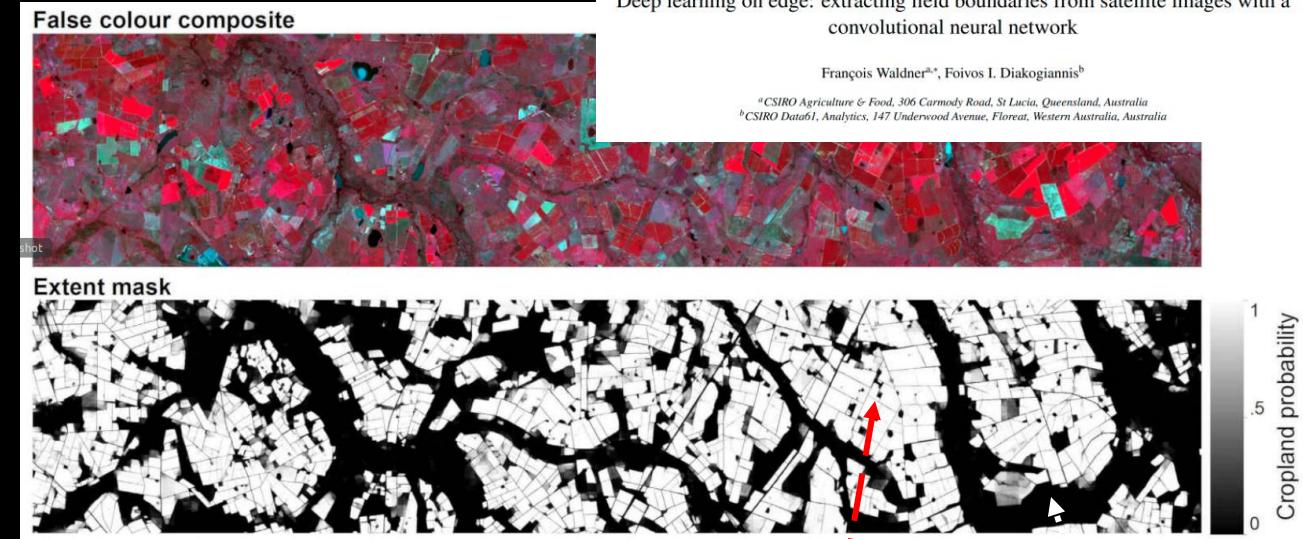
    att_spat = self.metric_space(q,k) # B,1,H,W
    v_spat = att_spat * v # emphasize spatial features

    att_chan = self.metric_channel(q,k) # B,C,1,1
    v_chan = att_chan * v # emphasize spatial features

    v_cspat = 0.5*(v_chan + v_spat) # emphasize spatial features
    v_cspat = self.norm(v_cspat)

    return v_cspat

```



Deep learning on edge: extracting field boundaries from satellite images with a convolutional neural network

François Waldner^{a,*}, Foivos I. Diakogiannis^b

^aCSIRO Agriculture & Food, 306 Carmody Road, St Lucia, Queensland, Australia

^bCSIRO Data61, Analytics, 147 Underwood Avenue, Floreat, Western Australia, Australia

Erases information where $A \rightarrow 0$

1s

0s

$$F = L + \gamma L \odot A = L \odot (1 + \gamma A)$$

Start without attention $\gamma_0 = 0$
(SAGAN)

Relative attention: how do you combine the FracTAL with two feature layers?

```

def forward(self, input1, input2, input3):

    # These should work with ReLU as well
    q = mx.npx.sigmoid(self.query(input1))
    k = mx.npx.sigmoid(self.key(input2))# B,C,H,W
    v = mx.npx.sigmoid(self.value(input3)) # B,C,H,W

    att_spat = self.metric_space(q,k) # B,1,H,W
    v_spat = att_spat * v # emphasize spatial features

    att_chan = self.metric_channel(q,k) # B,C,1,1
    v_chan = att_chan * v # emphasize spatial features

    v_cspat = 0.5*(v_chan + v_spat) # emphasize spatial features
    v_cspat = self.norm(v_cspat)

    return v_cspat

```

What is important on Layer 1 (resp. 2) based on
Information that exists on Layer 2 (resp. 1)?

$$\mathbf{F}_1 = \mathbf{L}_1 \odot [1 + \gamma_1 \mathbf{A}_{122}(\mathbf{q}(\mathbf{L}_1), \mathbf{k}(\mathbf{L}_2), \mathbf{v}(\mathbf{L}_2))] \quad (15)$$

$$\mathbf{F}_2 = \mathbf{L}_2 \odot [1 + \gamma_2 \mathbf{A}_{211}(\mathbf{q}(\mathbf{L}_2), \mathbf{k}(\mathbf{L}_1), \mathbf{v}(\mathbf{L}_1))] \quad (16)$$

$$\mathbf{F} = \text{Conv2DN}(\text{concat}([\mathbf{F}_1, \mathbf{F}_2])) \quad (17)$$

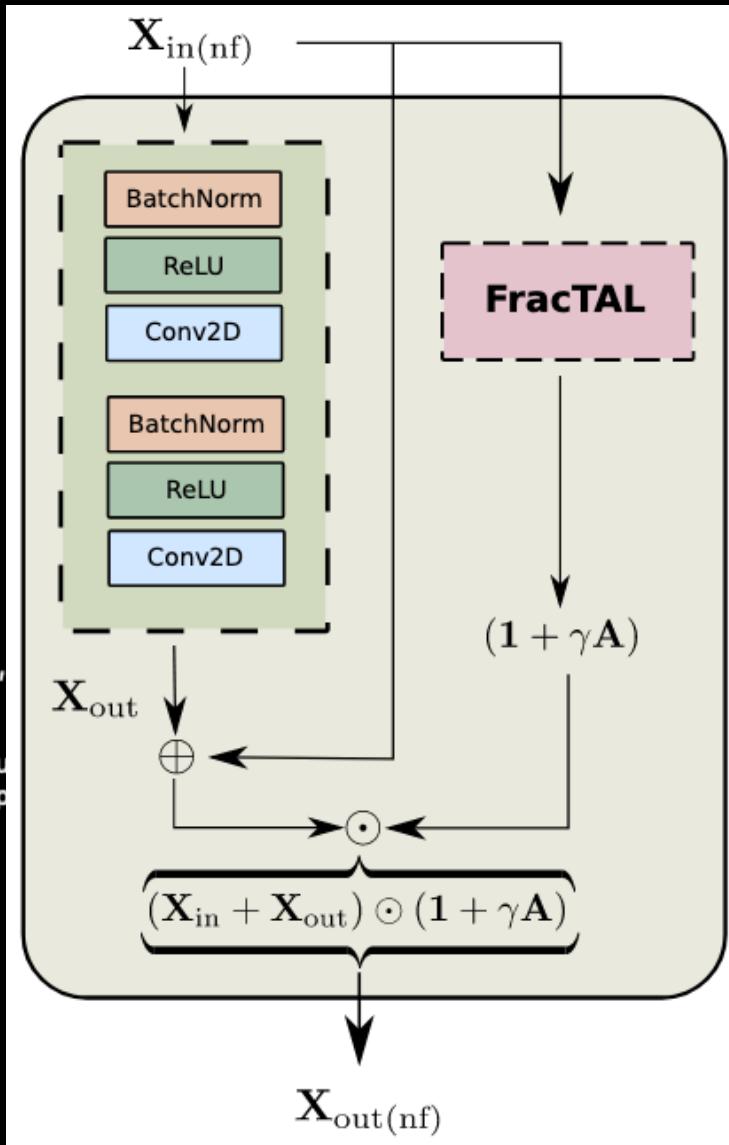
FracTAL ResNet (Robin) Based on self attention ideas

Robin

```

46 class FracTALResNet_unit(HybridBlock):
47     def __init__(self, nfilters, ngroups=1, nheads=1, kernel_size=(3,3), dilation_rate=(1,1), norm_type = 'BatchNorm',
48                  super().__init__(**kwargs)
49
50     self.block1 = ResNet_v2_block(nfilters,kernel_size,dilation_rate,_norm_type = norm_type, norm_groups=ngroups)
51     self.attn = FTAttention2D(nkeys=nfilters, nheads=nheads, kernel_size=kernel_size, norm = norm_type, norm_group=ngroups)
52
53     self.gamma = gluon.Parameter('gamma', shape=(1,), init=mx.init.Zero())
54
55     def forward(self, input):
56         out1 = self.block1(input)
57
58         att2 = self.attn(input)
59         att2 = self.gamma.data() * att2
60
61         out = (input + out1) * (mx.nd.ones_like(out1) + att2)
62
63         return out

```



CEECNet & FracTAL ResNet micro-topologies (units)

CEECNet - relative + self FracTAL

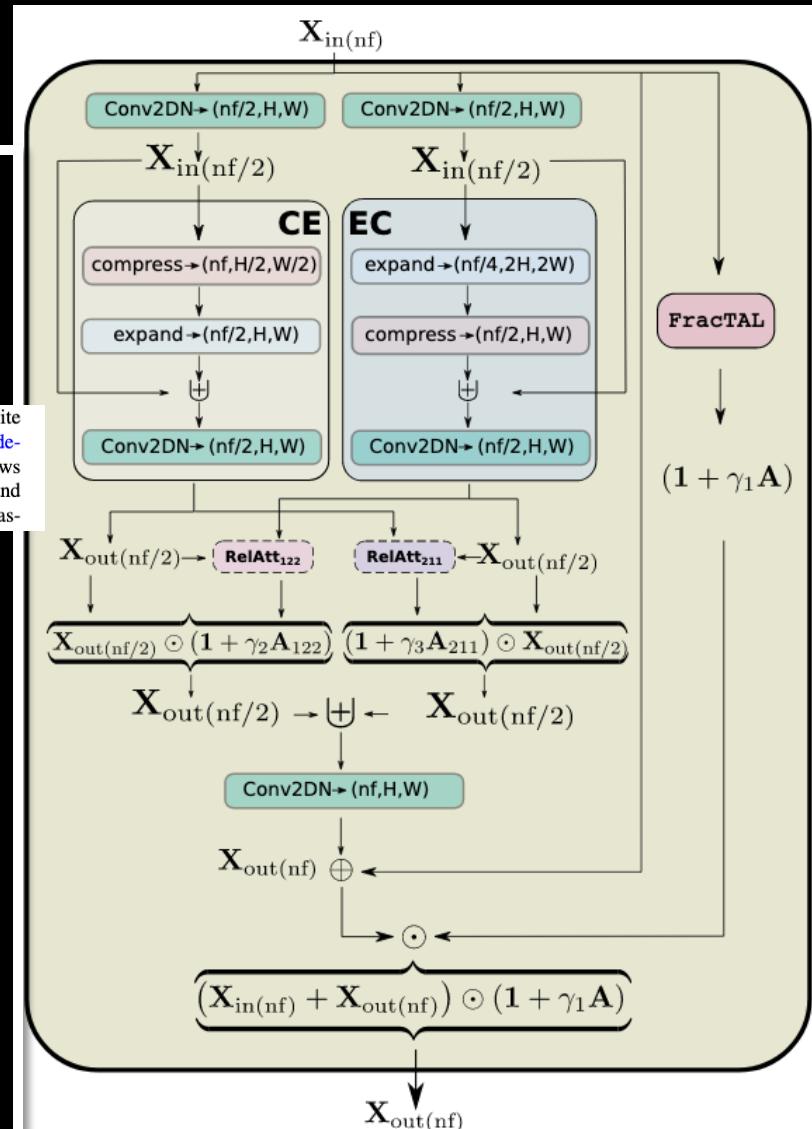
Batman

CEECNET - relative + self Fractal Batman

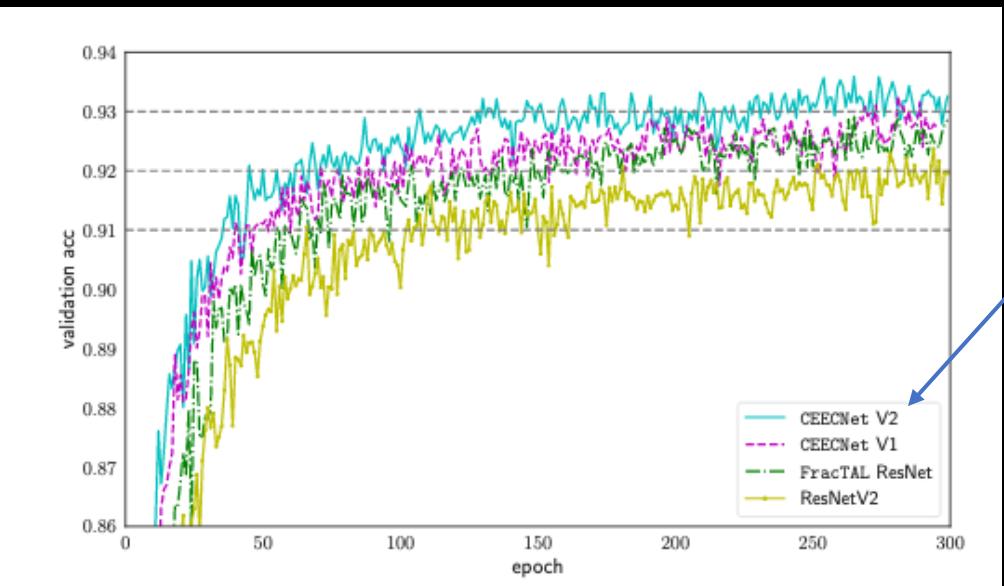
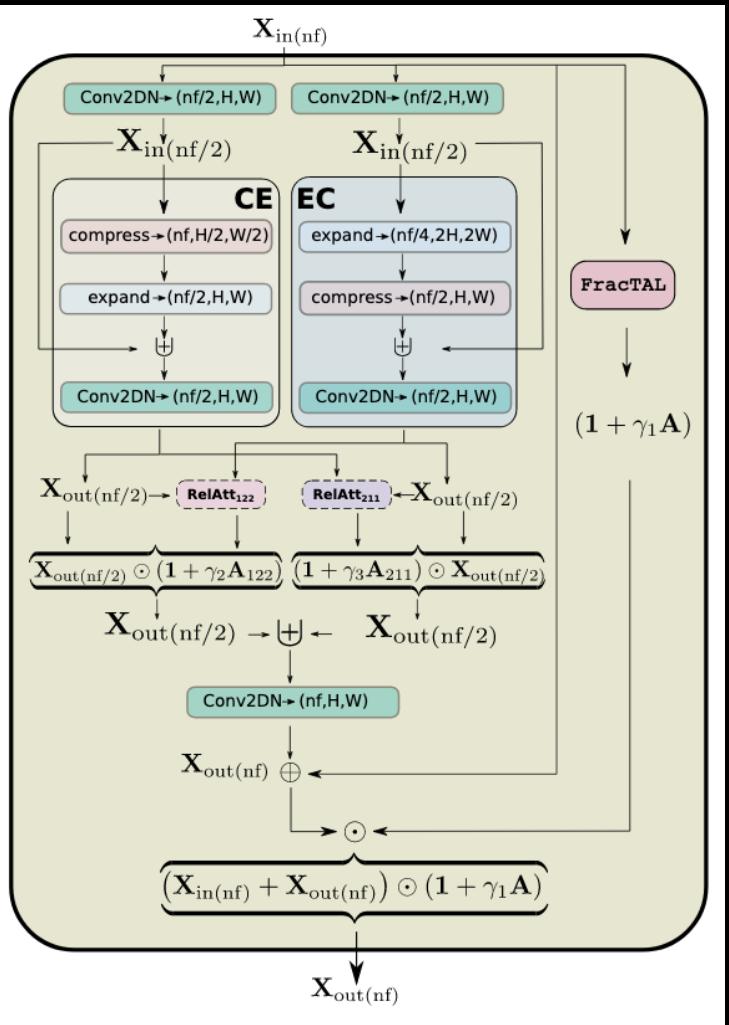
```

9 class CEEC_unit_v1(HybridBlock):
10     def __init__(self, nfilters, nheads=1, ngroups=1, norm_type='BatchNorm', norm_group=None, ftdepth=0, **kwargs):
11         super().__init__(**kwargs)
12
13
14         nfilters_init = nfilters//2
15         self.conv_init_1 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
16
17         self.compr1 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
18
19         self.compr2 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
20
21         self.expand1 = ExpandNCombine(nfilters_init, norm_type=norm_type, **kwargs)
22
23
24         self.size = self.conv_init_1.kernel_size[0]
25
26         self.conv_init_2 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
27
28         self.expand2 = ExpandLayer(nfilters_init//2, norm_type=norm_type, **kwargs)
29         self.compr21 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
30         self.compr22 = Conv2DNormed(channels=nfilters_init, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
31
32         # Will join with master input with concatenation -- IMPORTANT: ngrps
33         self.collect = Conv2DNormed(channels=nfilters, kernel_size=3, padding=1, strides=1, groups=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
34
35
36         self.att = FTAttention2D(nkeys=nfilters, nheads=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
37         self.ratt122 = RelFTAttention2D(nkeys=nfilters_init, nheads=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
38         self.ratt211 = RelFTAttention2D(nkeys=nfilters_init, nheads=nheads, norm_type=norm_type, norm_group=norm_group, ftdepth=ftdepth, **kwargs)
39
40
41         self.gamma1 = gluon.Parameter('gamma1', shape=(1,), init=mx.init.Zeros)
42         self.gamma2 = gluon.Parameter('gamma2', shape=(1,), init=mx.init.Zeros)
43         self.gamma3 = gluon.Parameter('gamma3', shape=(1,), init=mx.init.Zeros)
44
45
46     def forward(self, input):
47
48         # ===== UNet branch =====
49         out10 = self.conv_init_1(input)
50         out11 = self.compr1(out10)
51         out11 = mx.npx.relu(out11)
52         out11 = self.compr11(out11)
53         out11 = mx.npx.relu(out11)
54         out11 = self.expand1(out11, out10)
55         out11 = mx.npx.relu(out11)
56
57
58         # ===== \capNet branch =====
59         out20 = self.conv_init_2(input)
60         out21 = self.expand2(out20)
61         out21 = mx.npx.relu(out21)
62         out21 = self.compr21(out21)
63         out21 = mx.npx.relu(out21)
64         out21 = self.compr22(mx.np.concatenate([out21, out20], axis=1))
65         out21 = mx.npx.relu(out21)
66
67
68         att = self.gammai.data() * self.att(input)
69         ratt122 = self.gamma2.data() * self.ratt122(out11, out21, out2)
70         ratt211 = self.gamma3.data() * self.ratt211(out21, out11, out1)
71
72
73         ones1 = mx.np.ones_like(out10)
74         ones2 = mx.np.ones_like(input)
75
76
77         # Enhanced output of 1, based on memory of 2
78         out122 = out11 * (ones1 + ratt122)
79         # Enhanced output of 2, based on memory of 1
80         out211 = out21 * (ones1 + ratt211)
81
82
83         out12 = mx.npx.relu(self.collect(mx.np.concatenate([out122, out211], axis=1)))
84
85
86         # Emphasize residual output from memory on input
87         out_res = (input + out12) * (ones2 + att)
88
89         return out_res
90
91
92
93
94
95
96
97
98
99
100
101
102
103     def forward(self, input):
104
105         # ===== UNet branch =====
106         out10 = self.conv_init_1(input)
107         out11 = self.compr1(out10)
108         out11 = mx.npx.relu(out11)
109         out11 = self.compr21(out11)
110         out11 = mx.npx.relu(out11)
111         out11 = self.expand1(out11, out10)
112         out11 = mx.npx.relu(out11)
113
114
115         # ===== \capNet branch =====
116
117         out20 = self.conv_init_2(input)
118         out21 = self.expand2(out20)
119         out21 = mx.npx.relu(out21)
120         out21 = self.compr21(out21)
121         out21 = mx.npx.relu(out21)
122         out21 = self.compr22(mx.np.concatenate([out21, out20], axis=1))
123         out21 = mx.npx.relu(out21)
124
125
126         att = self.gammai.data() * self.att(input)
127         ratt122 = self.gamma2.data() * self.ratt122(out11, out21, out2)
128         ratt211 = self.gamma3.data() * self.ratt211(out21, out11, out1)
129
130
131         ones1 = mx.np.ones_like(out10)
132         ones2 = mx.np.ones_like(input)
133
134
135         # Enhanced output of 1, based on memory of 2
136         out122 = out11 * (ones1 + ratt122)
137         # Enhanced output of 2, based on memory of 1
138         out211 = out21 * (ones1 + ratt211)
139
140
141         out12 = mx.npx.relu(self.collect(mx.np.concatenate([out122, out211], axis=1)))
142
143
144         # Emphasize residual output from memory on input
145         out_res = (input + out12) * (ones2 + att)
146
147         return out_res
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
```

Our hypothesis and motivation for this approach is quite similar to the scale-space analysis in computer vision (Lindeberg, 1994): viewing input features at different scales, allows the algorithm to focus on different aspects of the inputs, and thus perform more efficiently. The fact that by merely increasing



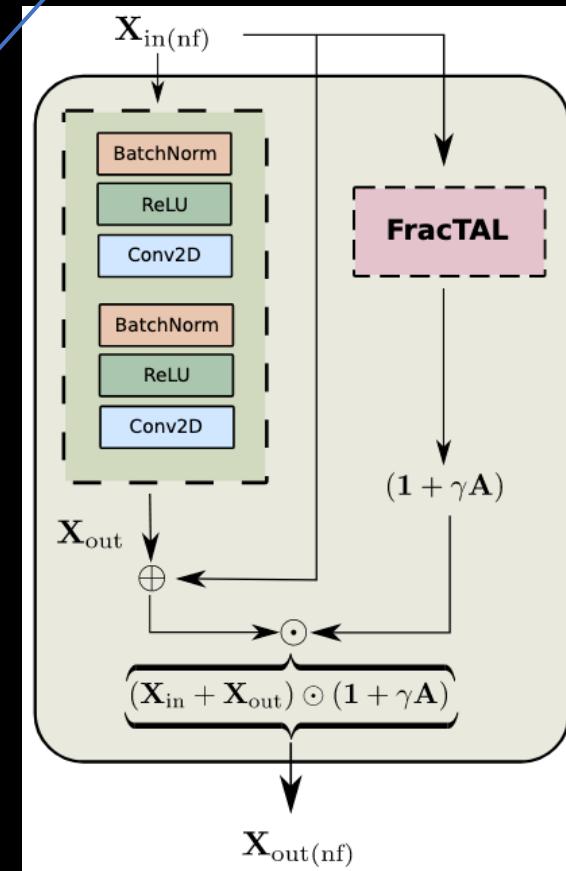
CEECNet & FracTAL ResNet (cifar10 test)



Batman

Robin

(there exist also a Hulk...)



How do we consume/compare two inputs? The mantis macro-topology

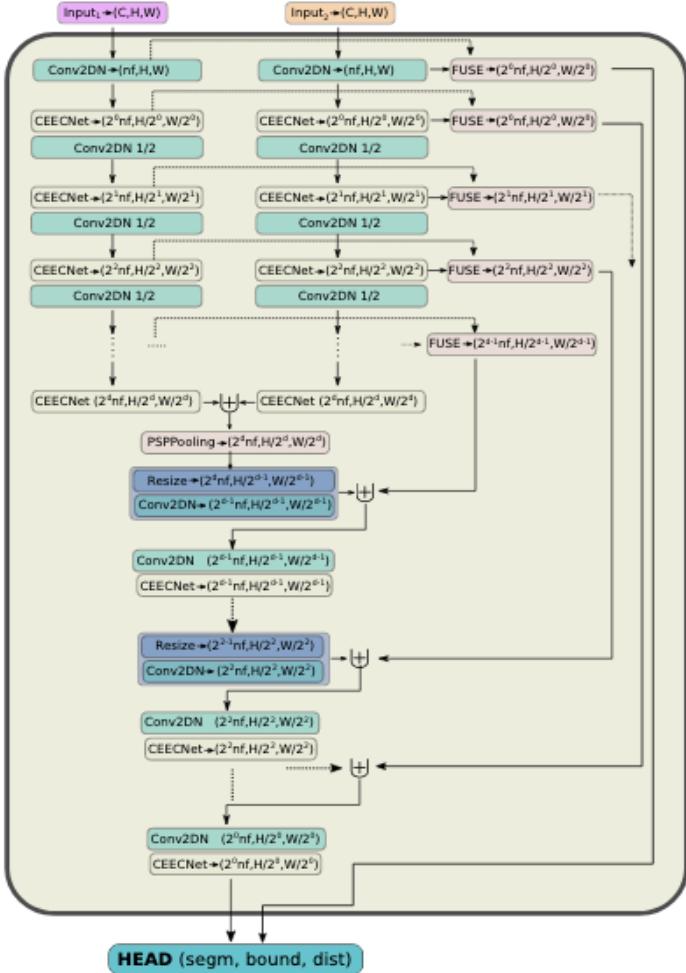


Figure 6: The mantis CEECNetV1 architecture for the task of change detection. The Fusion operation (FUSE) is described with MXNET/GLUON style pseudocode in detail on Listing 3.

Inductive bias: What is important on image 1 (resp. based on Information that exists on image 2 (resp.)

$$\mathbf{F}_1 = \mathbf{L}_1 \odot [\mathbf{1} + \gamma_1 \mathbf{A}_{122}(\mathbf{q}(\mathbf{L}_1), \mathbf{k}(\mathbf{L}_2), \mathbf{v}(\mathbf{L}_2))] \quad (15)$$

$$\mathbf{F}_2 = \mathbf{L}_2 \odot [\mathbf{1} + \gamma_2 \mathbf{A}_{211}(\mathbf{q}(\mathbf{L}_2), \mathbf{k}(\mathbf{L}_1), \mathbf{v}(\mathbf{L}_1))] \quad (16)$$

$$\mathbf{F} = \text{Conv2DN}(\text{concat}([\mathbf{F}_1, \mathbf{F}_2])) \quad (17)$$



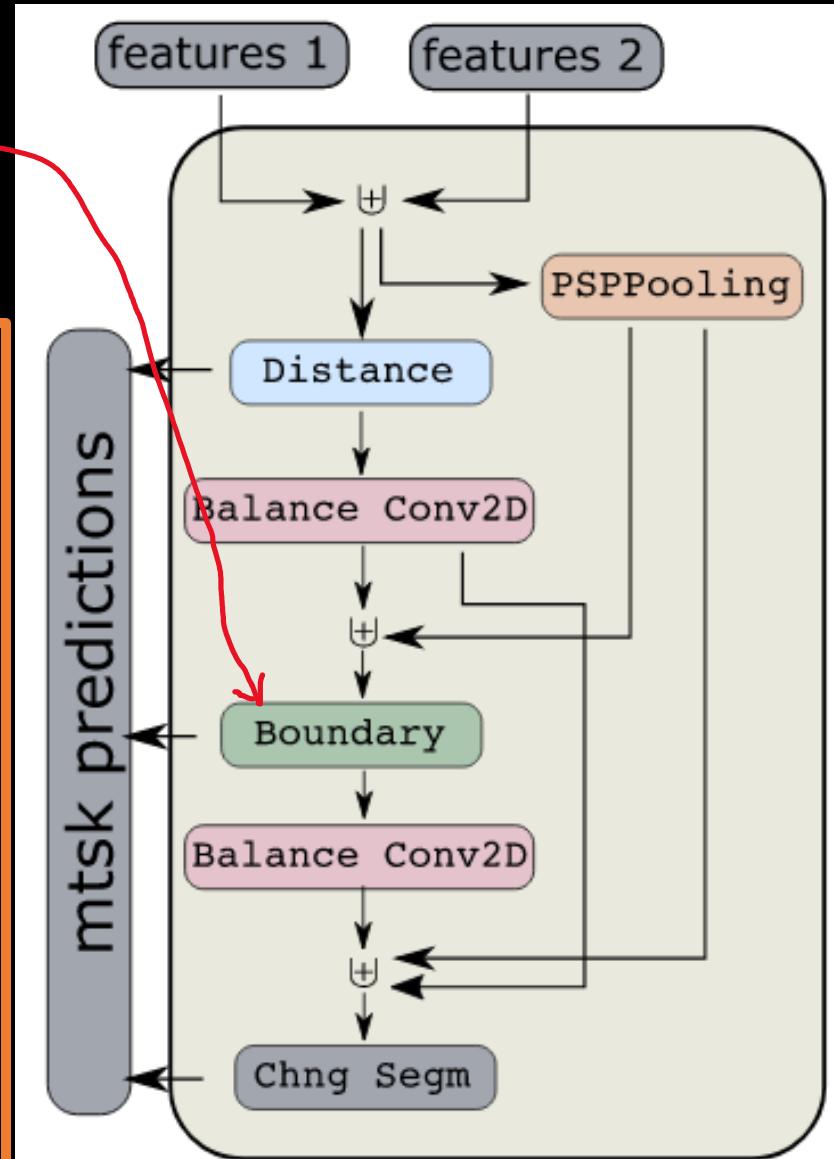
Improved (balanced) “causal” multitasking head + boundaries

```

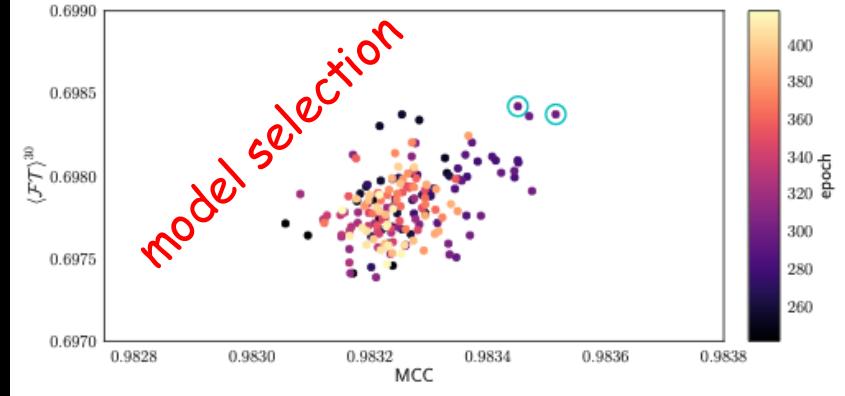
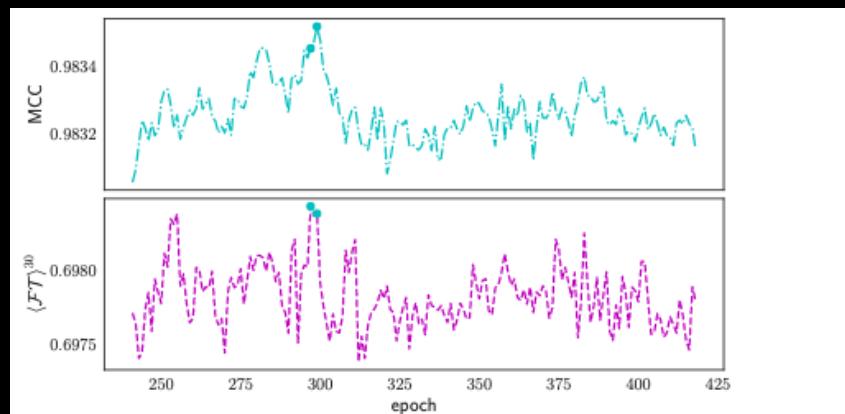
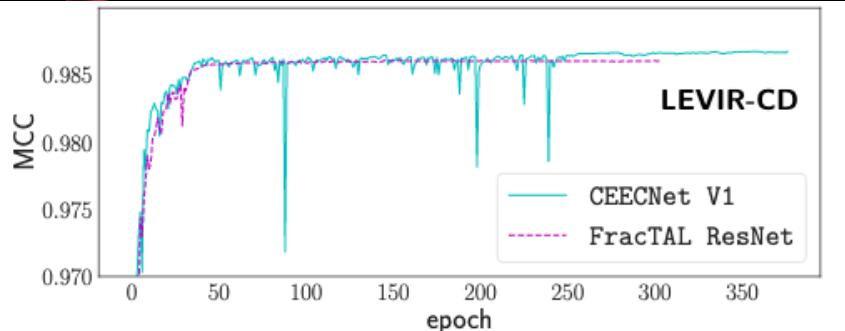
29 class Head_CMTSK_BC(HybridBlock):
30     # BC: Balanced (features) Crisp (boundaries)
31     def __init__(self, nfilters, NClasses, norm_type='batchnorm', norm_groups=None, **kwargs):
32         super().__init__(**kwargs)
33
34         self.model_name = "Head_CMTSK_BC"
35
36         self.nfilters = nfilters # Initial number of filters
37         self.NClasses = NClasses
38         self.psp_2ndlast = PSP_Pooling(self.nfilters, _norm_type=norm_type, norm_groups=norm_groups)
39
40         # bound logits
41         self.bound_logits = HeadSingle(self.nfilters, self.NClasses, norm_type=norm_type, norm_groups=norm_groups)
42         self.bound_Equalizer = Conv2DNormed(channels = self.nfilters, kernel_size = 1, _norm_type=norm_type, norm_groups=norm_groups)
43
44         # distance logits -- deeper for better reconstruction
45         self.distance_logits = HeadSingle(self.nfilters, self.NClasses, norm_type=norm_type, norm_groups=norm_groups)
46         self.dist_Equalizer = Conv2DNormed(channels = self.nfilters, kernel_size = 1, _norm_type=norm_type, norm_groups=norm_groups)
47
48         self.Comb_bound = None
49         self.ChannelAct = None
50
51         # Segmenetation
52         self.final_segm = None
53
54
55         self.CrispSigm = None
56
57         # Last activation
58         if (self.NClasses > 1):
59             self.ChannelAct = nn.Softmax(dim=1)
60         else:
61             self.ChannelAct = nn.Sigmoid()
62
63         # XXX add skip connection here
64         self.ChannelAct = nn.Sequential(*[self.ChannelAct])
65
66     def forward(self, UpConv4, conv1):
67         # second last layer
68         convl = mx.np.concatenate([conv1,UpConv4],axis=1)
69         conv = self.psp_2ndlast(convl)
70         conv = mx.npx.relu(conv)
71
72         # logits
73
74         # 1st find distance map, skeleton like, topology info
75         dist = self.distance_logits(convl) # do not use max pooling for distance
76         dist = self.ChannelAct(dist)
77         distEq = mx.npx.relu(self.dist_Equalizer(dist)) # makes nfilters equals to conv and convl
78
79
80         # Then find boundaries
81         bound = mx.np.concatenate([conv, distEq],axis=1)
82         bound = self.bound_logits(bound)
83         bound = self.CrispSigm(bound) # Boundaries are not mutually exclusive
84         boundEq = mx.npx.relu(self.bound_Equalizer(bound))
85
86
87         # Now combine all predictions in a final segmentation mask
88         # Balance first boundary and distance transform, with the features
89         comb_bd = self.Comb_bound_dist(mx.np.concatenate([boundEq, distEq],axis=1))
90         comb_bd = mx.npx.relu(comb_bd)
91
92
93         all_layers = mx.np.concatenate([comb_bd, conv],axis=1)
94         final_segm = self.final_segm_logits(all_layers)
95         final_segm = self.ChannelAct(final_segm)
96
97
98         return final_segm, bound, dist

```

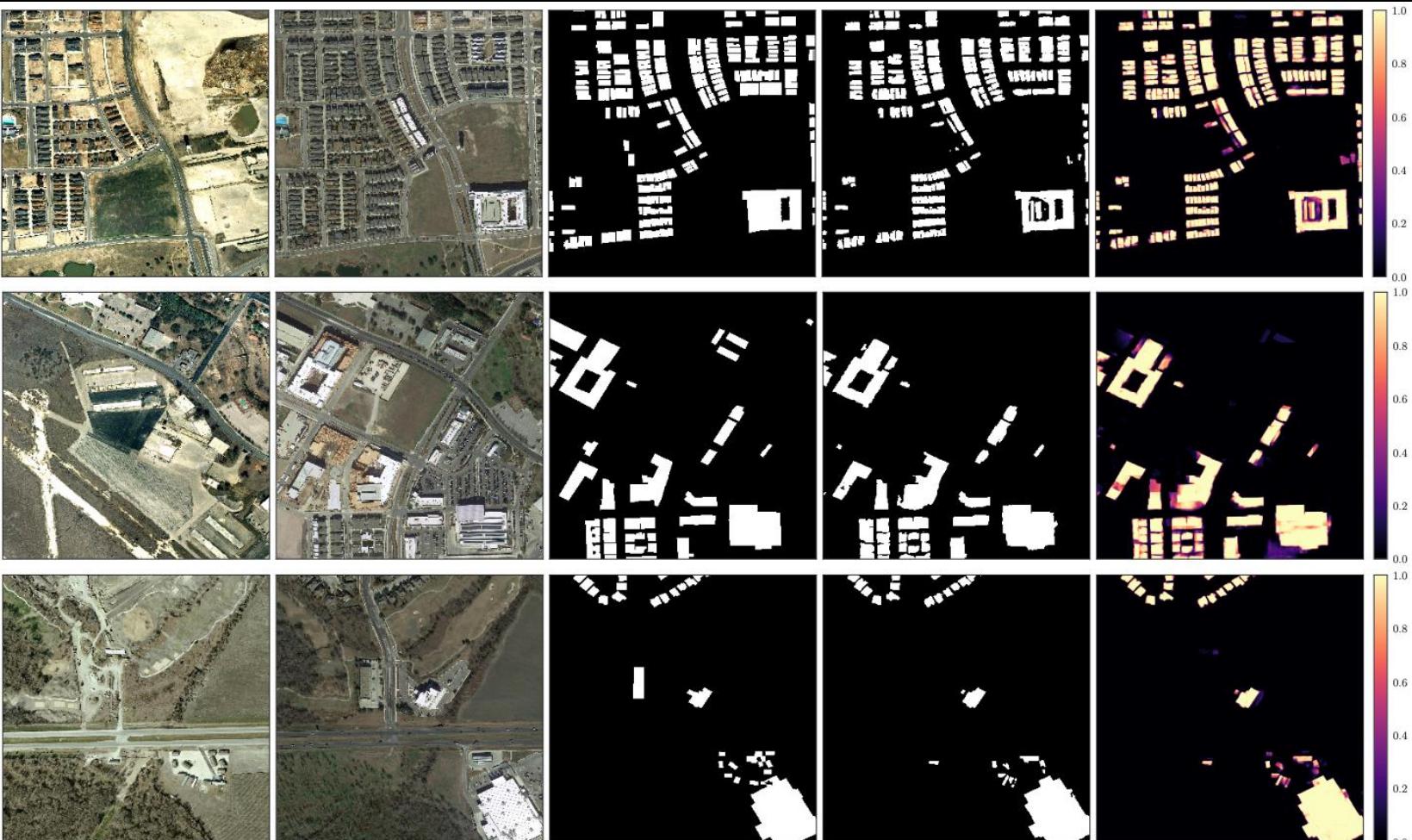
$$\text{sigmoid}_{\text{crisp}}(x) = \text{sigmoid}(x/\gamma), \quad \gamma \in [\epsilon, 1]$$



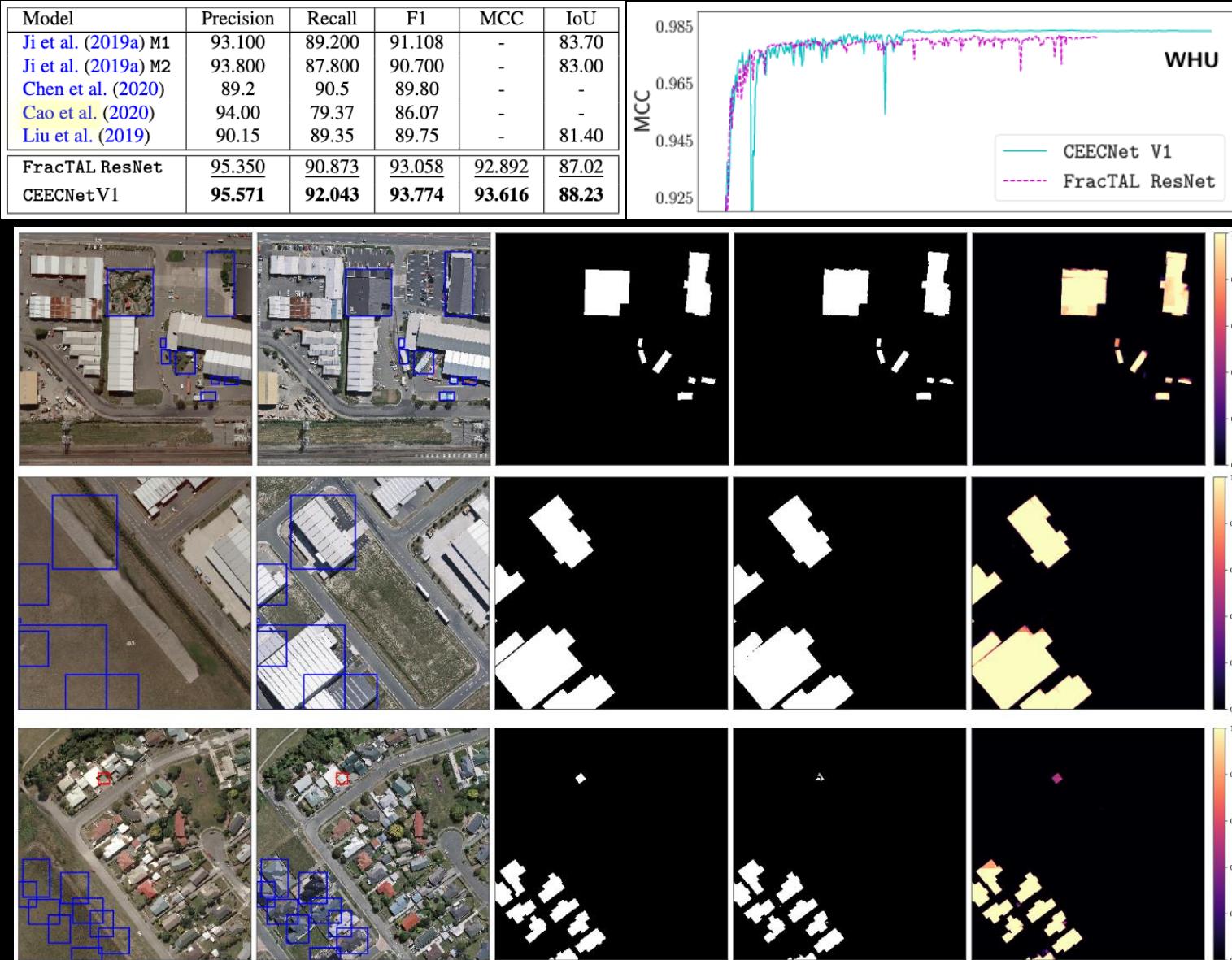
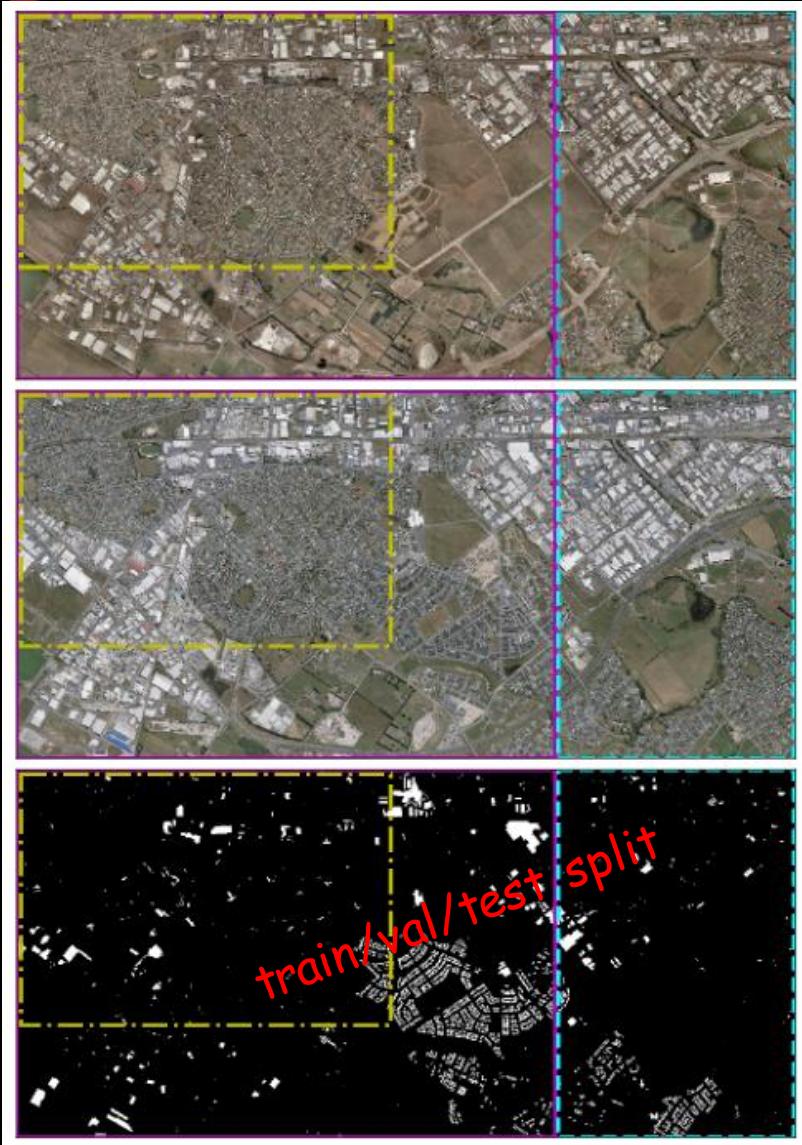
Change detection: D6nf32 performance on LEVIRCD (achieves SOTA)



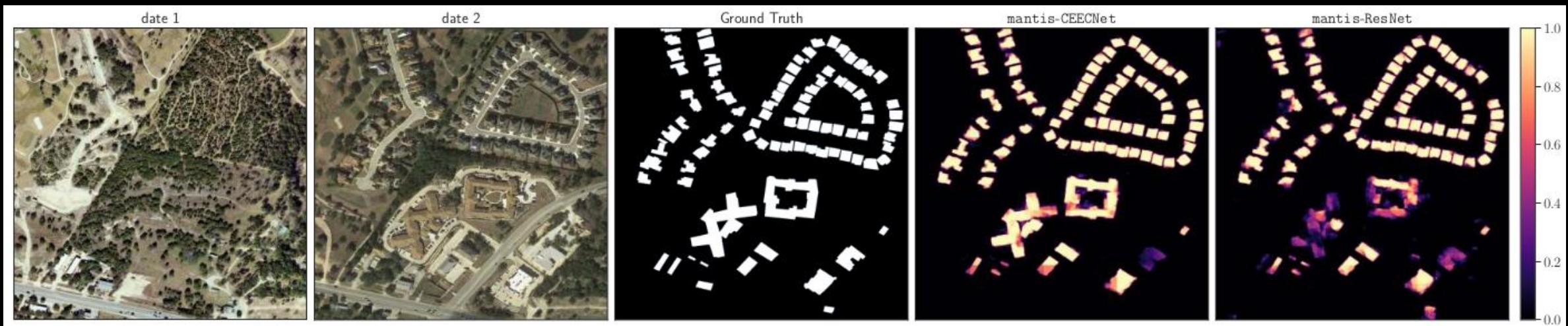
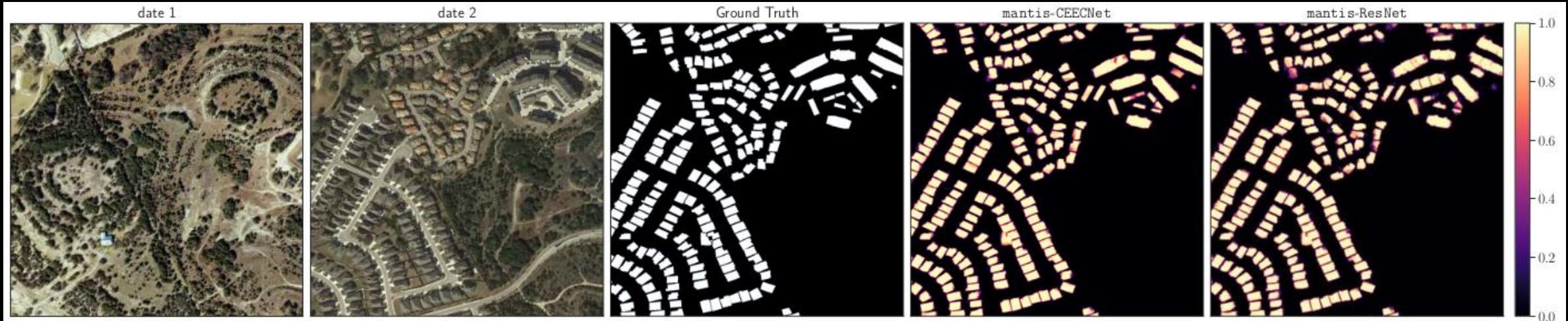
Model	Precision	Recall	F1	MCC	IoU
Chen and Shi (2020)	83.80	91.00	87.30	-	-
FracTAL ResNet	93.60	89.38	91.44	91.02	84.23
CEECNetV1	93.73	<u>89.93</u>	91.79	91.38	84.82



Change detection: D6nf32 performance on WHU (achieves SOTA*)



CEECNet V1 vs FracTAL ResNet performance



Not just change: Fusion

The Fusion method is **generic**: not just for change detection!

Data Fusion of two strongly **correlated** types of **inputs**

(replace all your concat operations)

Inductive bias: What is important on “image” 1 (resp. 2) based on Information that exists on “image” 2 (resp. 1)?

Relational inductive biases, deep learning, and graph networks

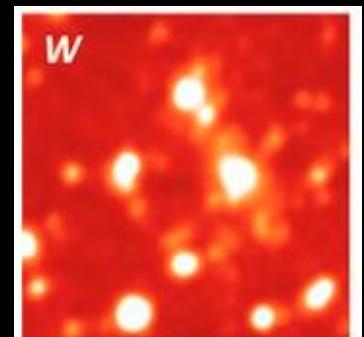
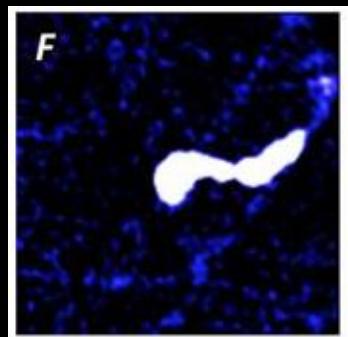
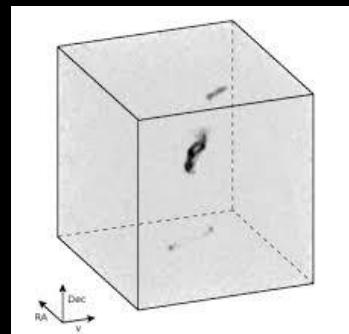
Peter W. Battaglia^{1*}, Jessica B. Hamrick¹, Victor Bapst¹, Alvaro Sanchez-Gonzalez¹, Vinicius Zambaldi¹, Mateusz Malinowski¹, Andrea Tacchetti¹, David Raposo¹, Adam Santoro¹, Ryan Faulkner¹, Caglar Gulcehre¹, Francis Song¹, Andrew Ballard¹, Justin Gilmer², George Dahl², Ashish Vaswani², Kelsey Allen³, Charles Nash⁴, Victoria Langston¹, Chris Dyer¹, Nicolas Heess¹, Daan Wierstra¹, Pushmeet Kohli¹, Matt Botvinick¹, Oriol Vinyals¹, Yujia Li¹, Razvan Pascanu¹

¹DeepMind; ²Google Brain; ³MIT; ⁴University of Edinburgh

$$\mathbf{F}_1 = \mathbf{L}_1 \odot [\mathbf{1} + \gamma_1 \mathbf{A}_{122}(\mathbf{q}(\mathbf{L}_1), \mathbf{k}(\mathbf{L}_2), \mathbf{v}(\mathbf{L}_2))] \quad (15)$$

$$\mathbf{F}_2 = \mathbf{L}_2 \odot [\mathbf{1} + \gamma_2 \mathbf{A}_{211}(\mathbf{q}(\mathbf{L}_2), \mathbf{k}(\mathbf{L}_1), \mathbf{v}(\mathbf{L}_1))] \quad (16)$$

$$\mathbf{F} = \text{Conv2DN}(\text{concat}([\mathbf{F}_1, \mathbf{F}_2])) \quad (17)$$



ClaRAN3D
(under construction)



Thank you

Interesting DL problem
(1D/2D/3D)?

Talk to us!

foivos.diakogiannis@icrar.org

