Semi-Implicit Convolutional Neural Networks

Eldad Haber[1 2], Keegan Lensink[* 1 2], Lars Ruthotto[3], Eran Triester[4]

[*]Presenter.
[1] Department of Earth Ocean and Atmospheric Sciences, The University of British Columbia, Vancouver, Canada.
[2] Xtract AI, Vancouver, Canada.
[3] Department of Mathematics and Computer Science, Emory University, Atlanta, GA.
[4] Department of Computer Science, Ben Gurion University, Be’er Sheva, Israel

Outline

1) **Introduction**
    1) High vs Low Dimensional Output
    2) Receptive Field
    3) Neural Network Stability
2) **IMEXnet**
    1) Formulation
    2) Implementation
3) **Numerical Experiments**
    1) Q-tips
    2) NYU Depth

Classification

Low Dimensional Output

$\rightarrow$

Class	Probability
Truck	0.01
Mouse	0.85
Dog	0.05
Cat	0.09

$$R^{h \mathrm{x} w \mathrm{x} d} \quad \rightarrow \quad R^k$$

where h, w, and d are the image dimensions, and k is the number of target classes.

Semantic Segmentation

High Dimensional Output

$\rightarrow$

$$R^{h\mathrm{x}w\mathrm{x}d} \rightarrow R^{h\mathrm{x}w\mathrm{x}k}$$

where h, w, and d are the image dimensions, and k is the number of target classes.

Semantic Segmentation

High Dimensional Output

$\rightarrow$

$$R^{h\text{x}w\text{x}d} \rightarrow R^{h\text{x}w\text{x}k}$$

where h, w, and d are the image dimensions, and k is the number of target classes.

$$k \geq d$$

Receptive Field

IMEXnet

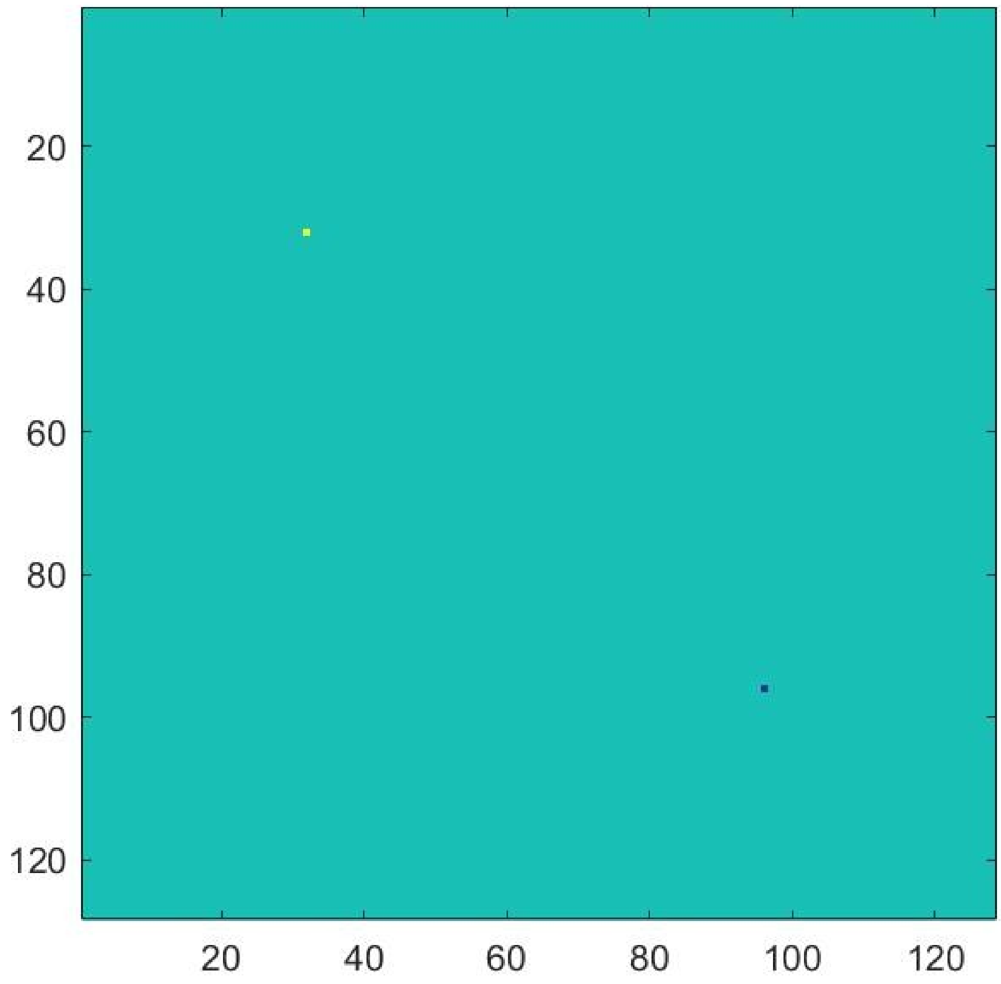
ResNet:

$$\mathbf{Y}^{n+1} = \mathbf{Y}^{n} + \mathbf{K}_2 \sigma(N_{\alpha,\beta}(\mathbf{K}_1 \mathbf{Y}^{n})) \quad \text{for} \quad n = 0, ..., N-1$$

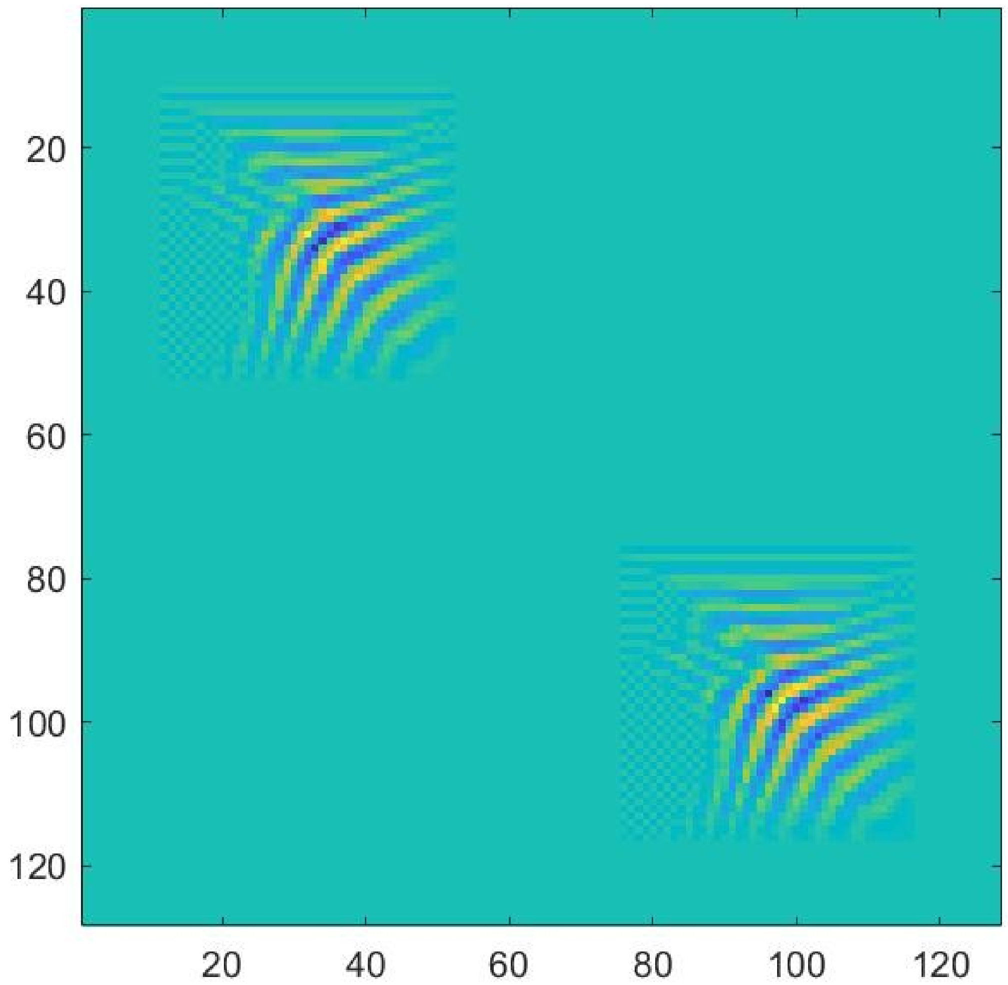
where $\mathbf{Y}^0$ is the input image, $\mathbf{K}_{1,2}$ are convolutions, σ is a non-linear activation function, and N is a normalization function that depends on α and β

K. He et al. (2015)

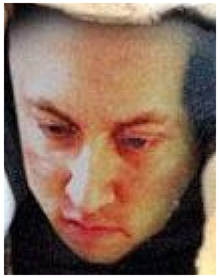
Receptive Field



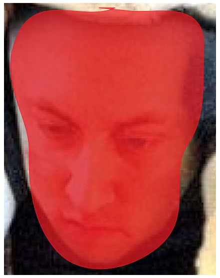
Receptive Field



Receptive Field



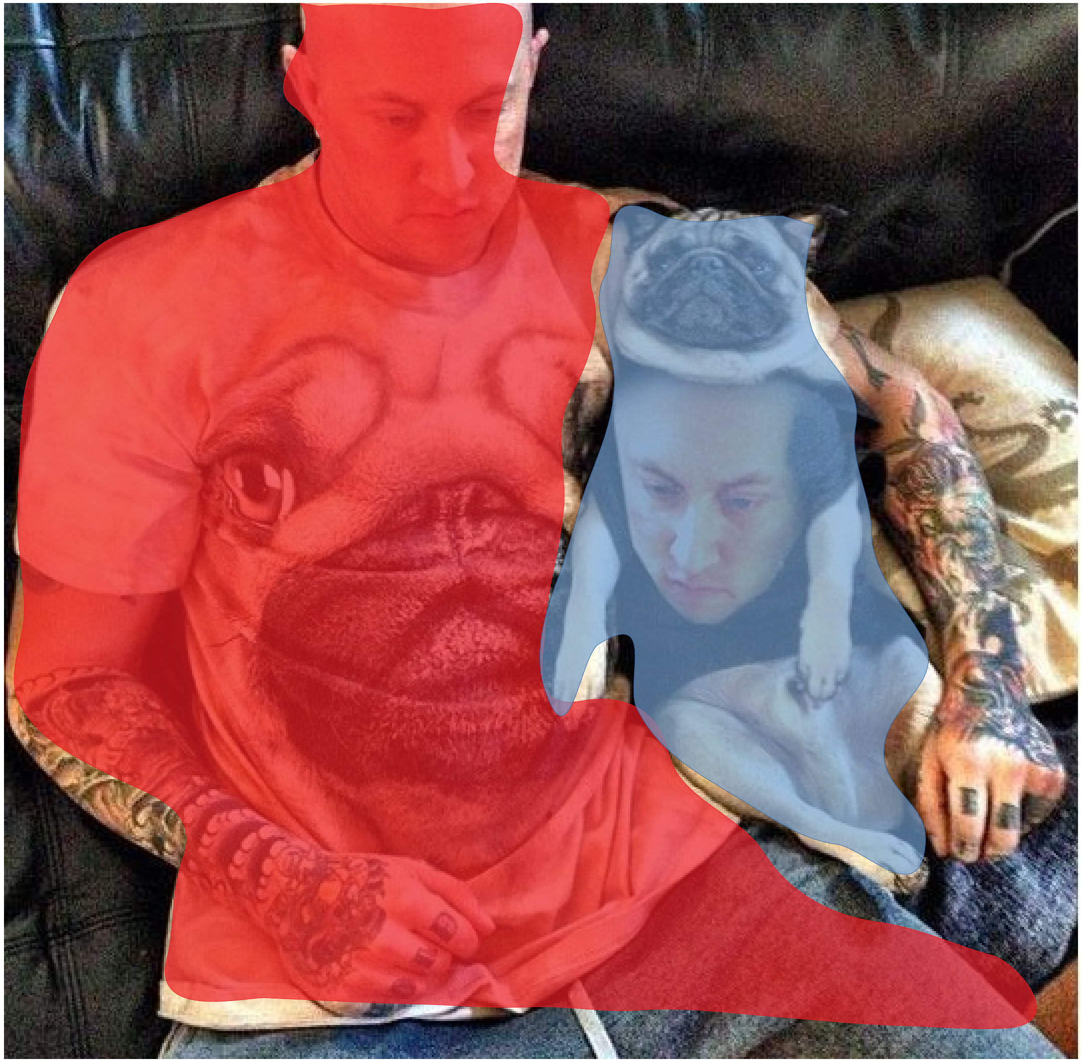
Receptive Field



Receptive Field

Non-local information is often necessary.

Receptive Field



Receptive Field

Receptive Field

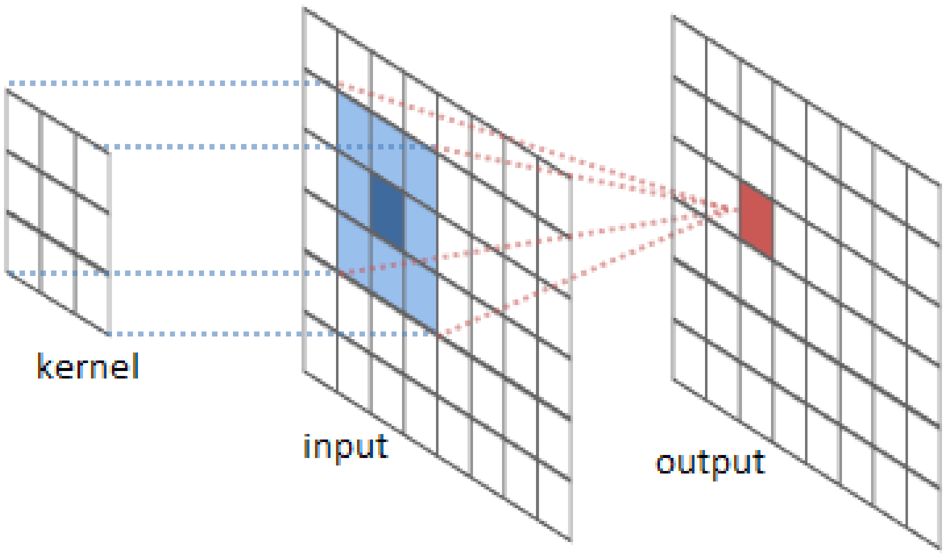
Receptive Field

How can we increase a network's receptive field?

Receptive Field

How can we increase a network's receptive field?

1) Convolutions



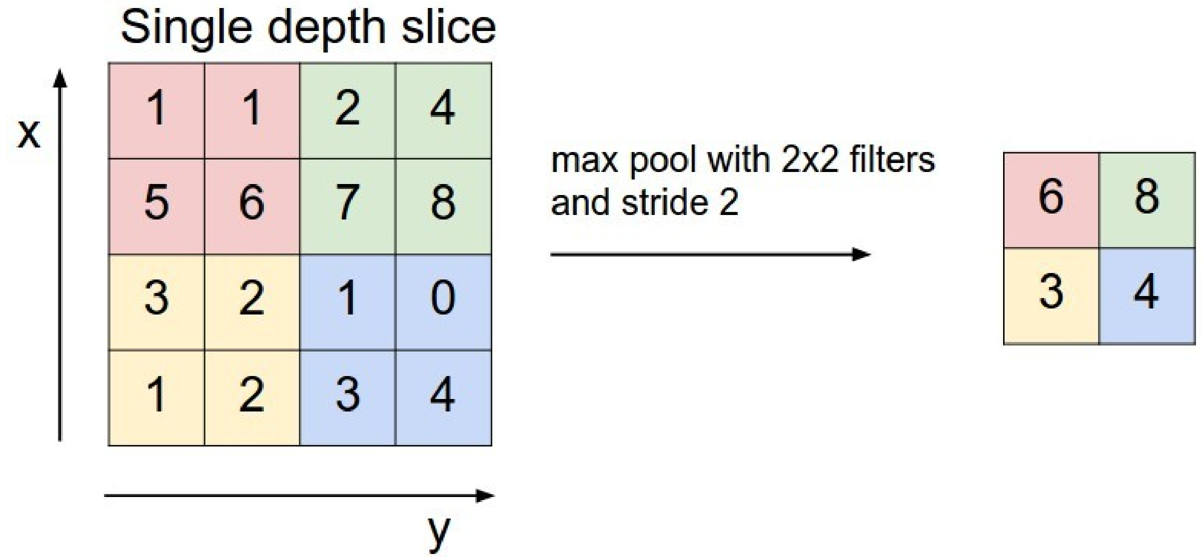
http://colah.github.io/posts/2014-07-Understanding-Convolutions/

Receptive Field

How can we increase a network's receptive field?

1) Convolutions

2) Coarsening

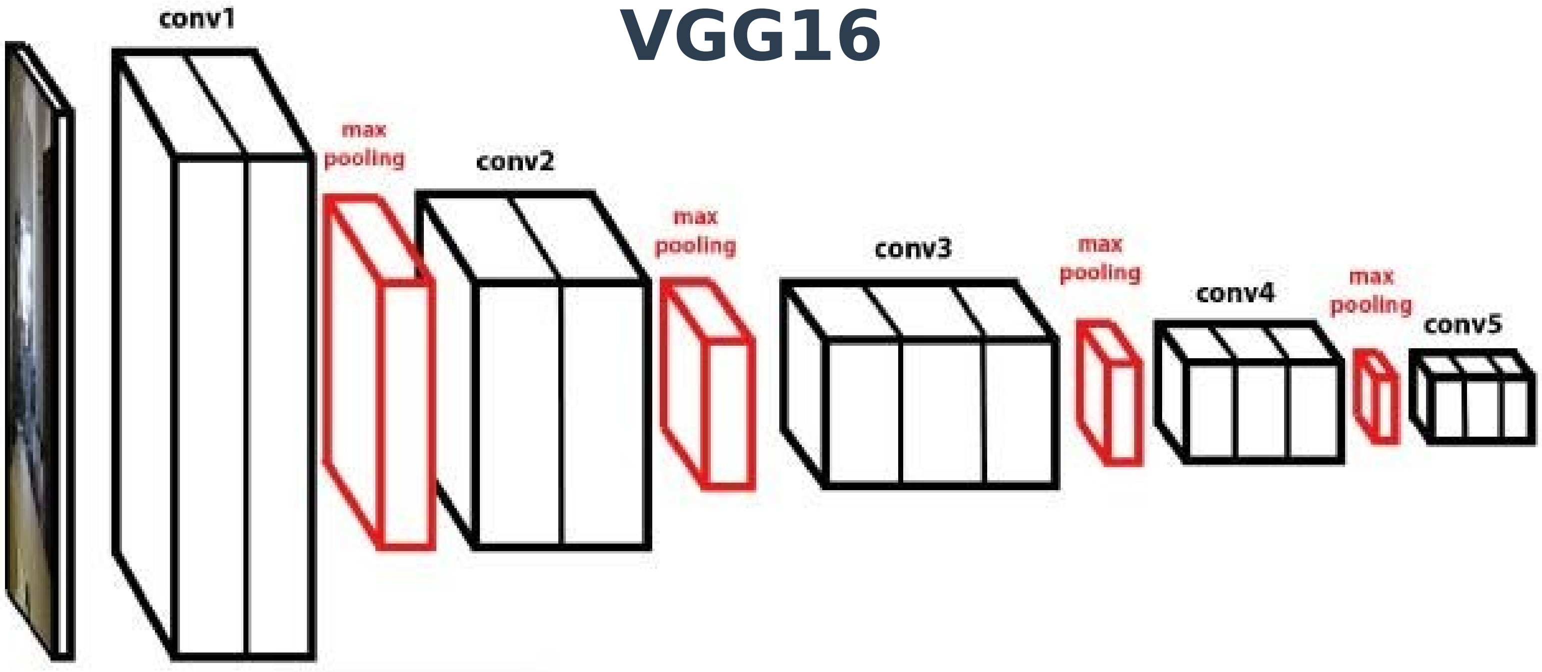


http://cs231n.github.io/convolutional-networks/

Receptive Field

In practice, a combination of both

VGG16

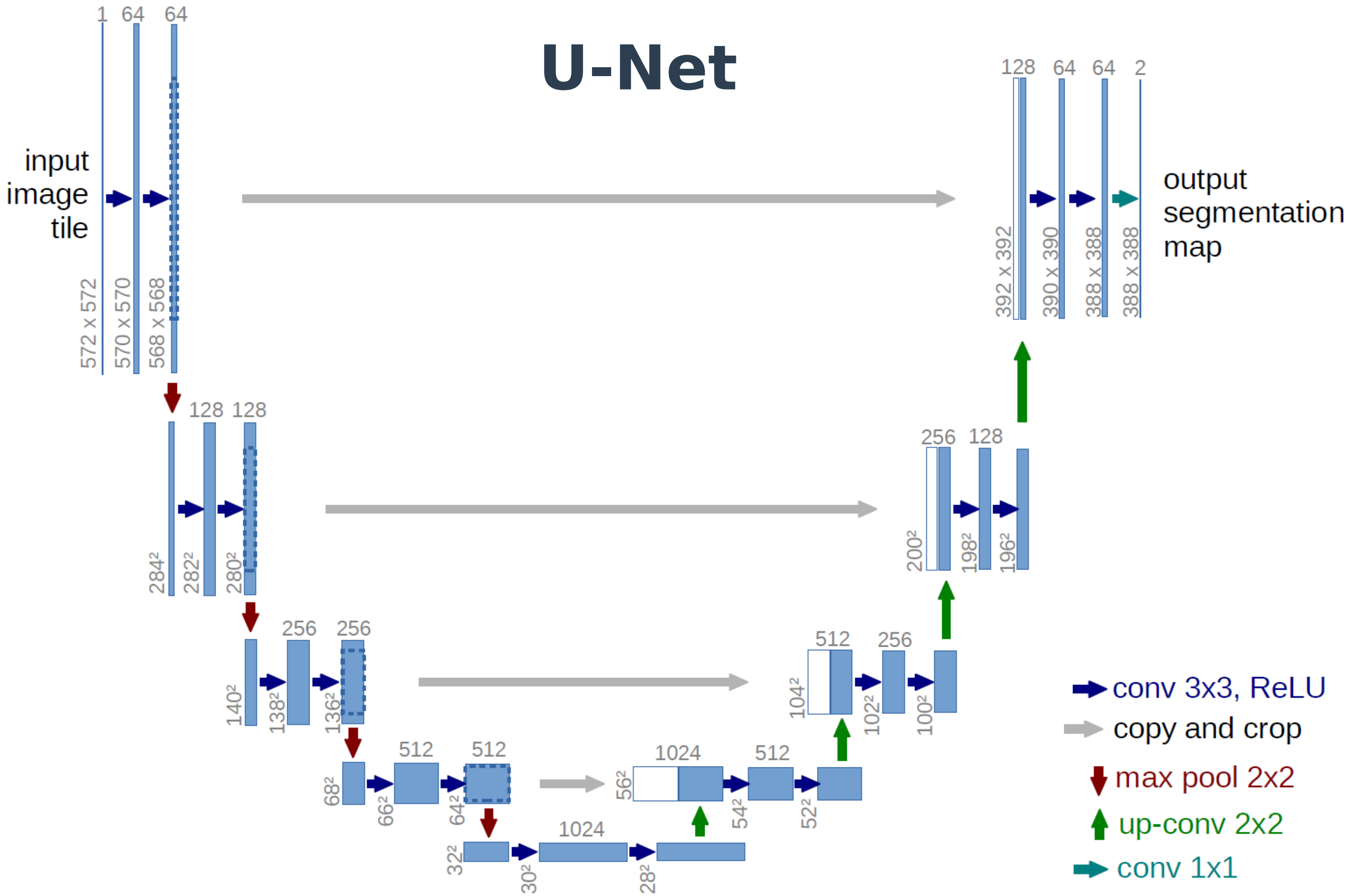


https://www.cs.toronto.edu/~frossard/post/vgg16/

Receptive Field

In practice, a combination of both

U-Net



O. Ronneberger and P.Fischer and T. Brox (2015)

Receptive Field

How can we increase a network’s receptive field?

1) Convolutions
   - Pros: Increase model complexity, increase receptive field
   - Cons: More expensive, less stable

2) Coarsening
   - Pros: Reduce dimensionality, increase receptive field
   - Cons: Lose information

Receptive Field

How can we increase a network’s receptive field?

1) Convolutions
   - Pros: Increase model complexity, increase receptive field
   - Cons: More expensive, less stable
2) Coarsening
   - Pros: Reduce dimensionality, increase receptive field
   - Cons: Lose information
3) Implicit Steps
   - Pros: Increase stability, cheap, couple all pixels
   - Cons: Additional hyperparameters, periodicity (depending on implementation)

Stability

Forward Stability:

- Small changes to the input result in small changes to the output

Stability

Forward Stability:

- Small changes to the input result in small changes to the output

- Small pertubations do not grow at depth

Stability

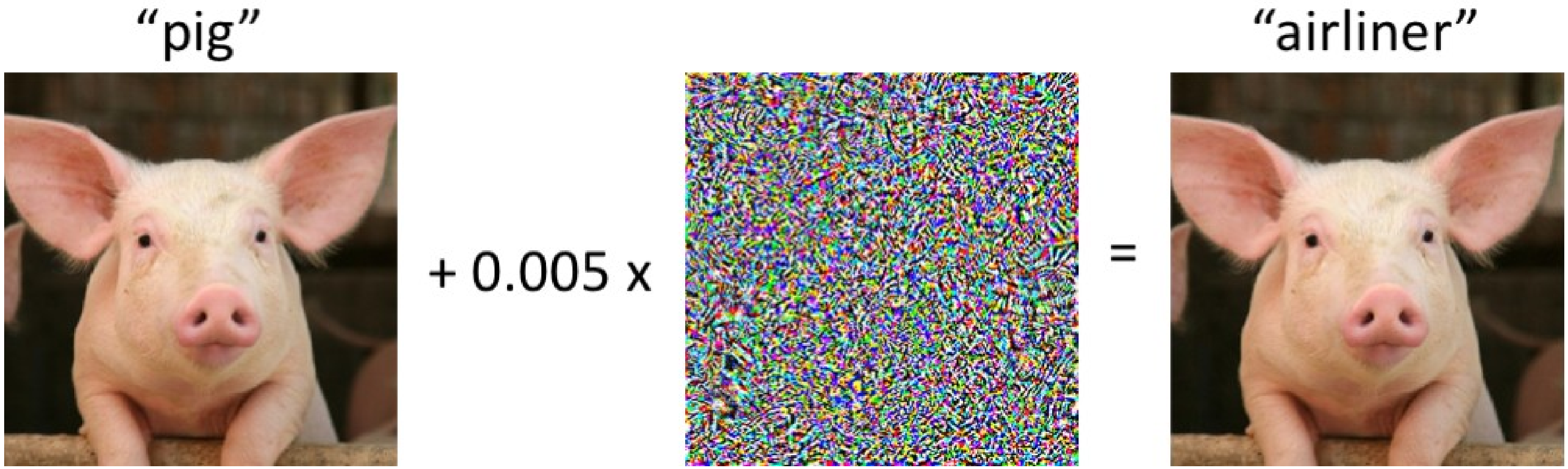
Forward Stability:

- Small changes to the input result in small changes to the output

- Small pertubations do not grow at depth, which could lead to vanishing/exploding gradients, making deep networks difficult to train

Adversarial Examples

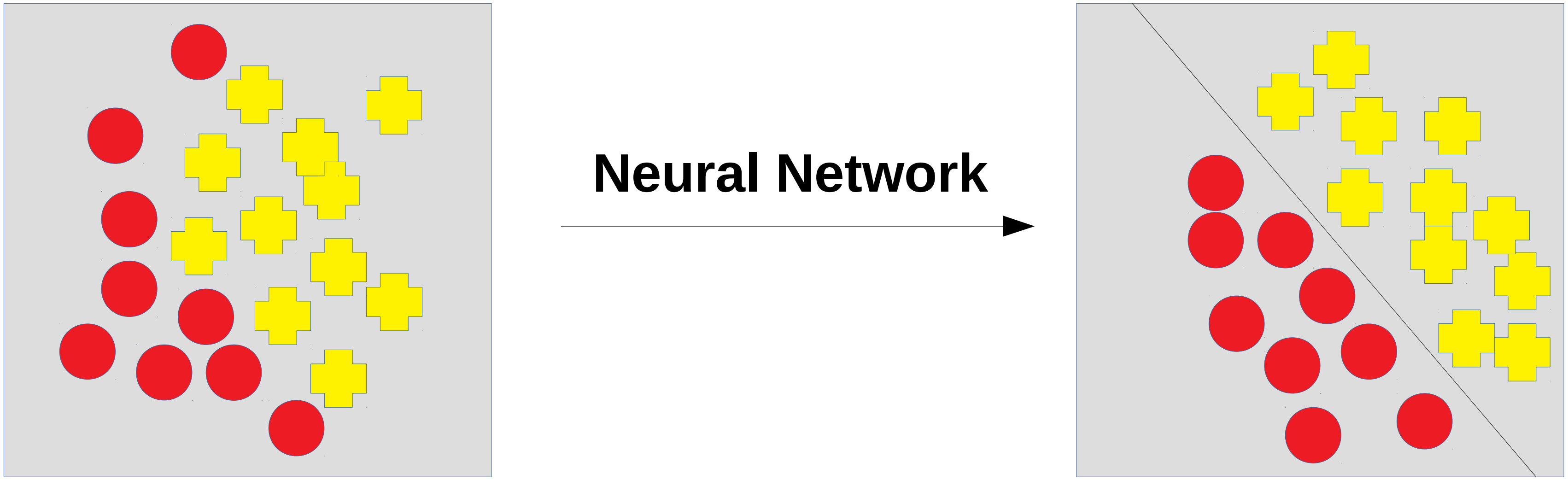
Exploit a network's instability inorder to fool a classifier



MIT CSAIL

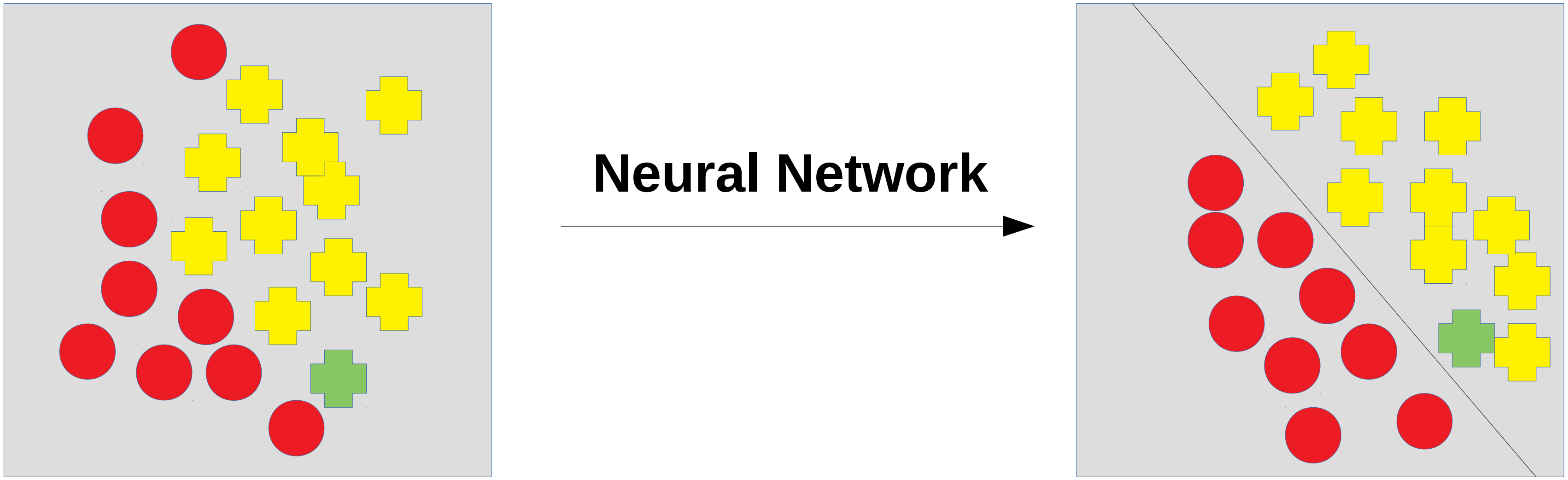
Adversarial Examples

Exploit a network's instability to fool a classifier



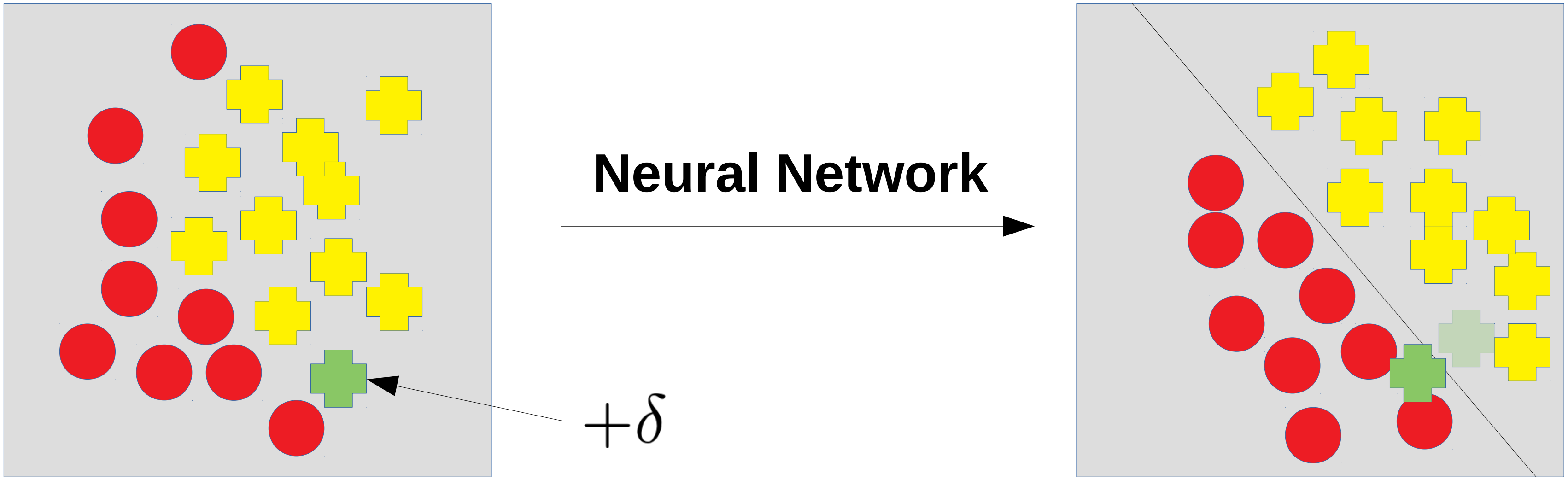
Adversarial Examples

Exploit a network's instability to fool a classifier



Adversarial Examples

Exploit a network's instability to fool a classifier



IMEXnet

Start with ResNet:

$$\mathbf{Y}^{n+1} = \mathbf{Y}^{n} + \mathbf{K}_2 \sigma(N_{\alpha,\beta}(\mathbf{K}_1 \mathbf{Y}^{n})) \quad \text{for} \quad n = 0, ..., N-1$$

where $\mathbf{Y}^0$ is the input image, $\mathbf{K}_{1,2}$ are convolutions, σ is a non-linear activation function, and N is a normalization function that depends on α and β

K. He et al. (2015)

IMEXnet

Start with ResNet:

$$\mathbf{Y}^{n+1} = \mathbf{Y}^n + f(\mathbf{Y}^n, \theta^n) \quad \text{for} \qquad n = 0, ..., N-1$$

Start with ResNet:

$$\mathbf{Y}^{n+1} = \mathbf{Y}^n + f(\mathbf{Y}^n, \theta^n) \quad \text{for} \quad n = 0, ..., N-1$$

Which can be seen with as a forward Euler discretization of the non-linear ODE (Haber & Ruthotto, 2017).

$$\dot{\mathbf{Y}}(\mathbf{t}) = f(\mathbf{Y}(t), \theta(t))$$

IMEXnet

We can view using a ResNet as solving the following initial value problem using forward Euler and a step size $h = 1$.

$$\dot{\mathbf{Y}}(\mathbf{t}) = f(\mathbf{Y}(t), \theta(t)), \quad \mathbf{Y}(0) = \mathbf{Y}_0$$

We are not limited to forward Euler, for example midpoint methods have been used, and RK methods have been proposed (Haber & Ruthotto, 2017; Chen et al., 2018).

IMEXnet

Recap of ResNet issues:

- Often unstable.
- Information takes many layers to travel across the computational grid.

One way to accelerate the communication of information across all pixels is to use an implicit method (Ascher & Petzold, 1998).

IMEXnet

Backward Euler

$$\mathbf{Y}^{n+1} = \mathbf{Y}^n + hf(\mathbf{Y}^{n+1}, \theta^{n+1}) \quad \text{for} \quad n = 0, ..., N-1$$

Treating the non-linear term implicilty requires solving the above equation at evey time step.

In order to avoid this expensive step, we instead consider an implicit-explicit (IMEX) method.

IMEX Method

Treat non-linear part of the RHS explicitly, and the linear part of it implicitly (Ascher et al, 1995; 1997).

Commonly used in fluid dynamics, surface formation, and image denoising.

IMEX Method

Treat non-linear part of the RHS explicitly, and the linear part of it implicitly.

There is no natural division in this context.

$$\dot{\mathbf{Y}}(t) = \underbrace{f(\mathbf{Y}(t), \boldsymbol{\theta}(t)) + \mathbf{LY}(t)}_{\text{explicit term}} - \underbrace{\mathbf{LY}(t)}_{\text{implicit term}}$$

where $\mathbf{L}$ is a linear invertible matrix

IMEX Method

We use forward Euler for the explicit terms, and backward Euler for the implicit term.

$$(\mathbf{I} + h\mathbf{L})\mathbf{Y}^{n+1} = \mathbf{Y}^{n}(\mathbf{I} + h\mathbf{L}) + hf(\mathbf{Y}^{n}, \theta^{n})$$

IMEX Method

We use forward Euler for the explicit terms, and backward Euler for the implicit term.

$$\mathbf{Y}^{n+1} = (\mathbf{I} + h\mathbf{L})^{-1}(\mathbf{Y}^n(\mathbf{I} + h\mathbf{L}) + hf(\mathbf{Y}^n, \theta^n))$$

IMEX Method

We use forward Euler for the explicit terms, and backward Euler for the implicit term.

$$\mathbf{Y}^{n+1} = (\mathbf{I} + h\mathbf{L})^{-1}(\mathbf{Y}^n(\mathbf{I} + h\mathbf{L}) + hf(\mathbf{Y}^n, \theta^n))$$

Note that the inverted term is dense, so it couples the entire computational grid in each step.

Absolute Stability

ResNet

$$\mathbf{Y}_{j+1} = (1 + h\lambda)\mathbf{Y}_j$$

$$\text{Stable if and only if } |1 + \lambda h| \leq 1$$

IMEXnet

$$\mathbf{Y}_{j+1} = \frac{1 + h\lambda + h\alpha}{1 + h\alpha}\mathbf{Y}_j \text{ when } \mathbf{L} = \alpha\mathbf{I}$$

$$\text{Stable if and only if } \left|\frac{1 + h\lambda + h\alpha}{1 + h\alpha}\right| \leq 1$$

Implementation

In the implicit step we solve the linear system

$$(\mathbf{I} + h\mathbf{L})\mathbf{Y} = \mathbf{B},$$

where **L** is a group convolution and **B** is the collection of explicit terms.

To compute this efficiently, we represent the convolution in the Fourier domain.

Implementation

Consider the convolution

$$\mathbf{A}\mathbf{Y} = \mathbf{B}, \ \text{where } \mathbf{A} = (\mathbf{I} + h\mathbf{L}).$$

By choosing **L** to be a convolution, we can use it's form in the Fourier domain

$$\mathbf{A} * \mathbf{Y} = \mathbf{F}^{-1}((\mathbf{F}\mathbf{A}) \odot (\mathbf{F}\mathbf{Y}))$$

to compute the product of the inverse

$$\mathbf{A}^{-1} * \mathbf{Y} = \mathbf{F}^{-1}((\mathbf{F}\mathbf{Y}) \oslash (\mathbf{F}\mathbf{A}))$$

$$\text{where } \oslash \text{ is element wise division.}$$

Implementation

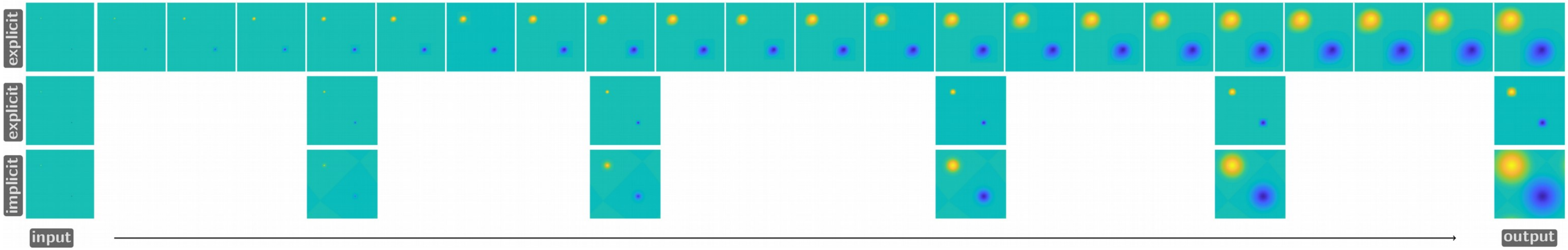
In order to ensure that **A** is invertible, we define **L** to be positive semi-definite in the form

$$\mathbf{L} = \mathbf{C}^T \mathbf{C},$$

where **C** is a groupwise convolution operator.

This implementation is CUDA enabled and supported by AutoGrad packages.

IMEXnet



Numerical Experiments

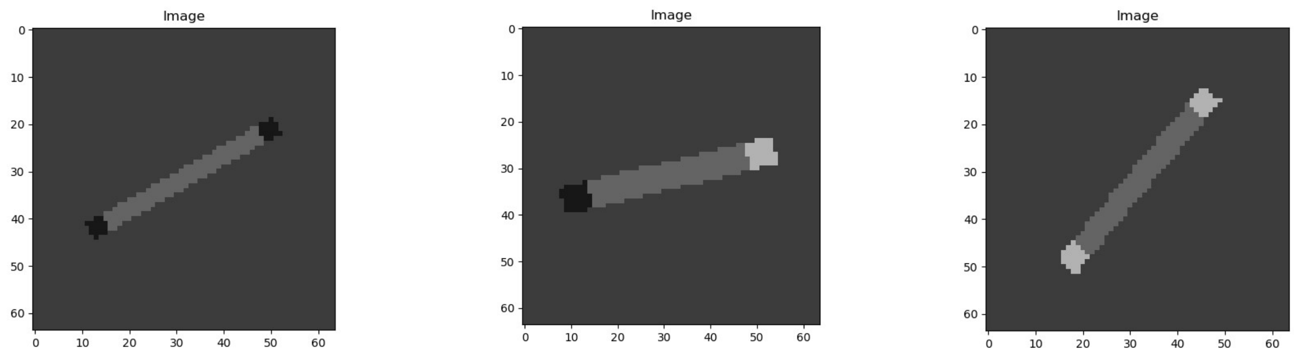
Q-tips Dataset

- Sythentic semantic segmentation dataset
- 1024 training examples, 64 valdiation examples
- Single object images randomly sampled from a unifrom distribution of lengths, widths, and orientations.
- 3 object classes
- Requires a large receptive field

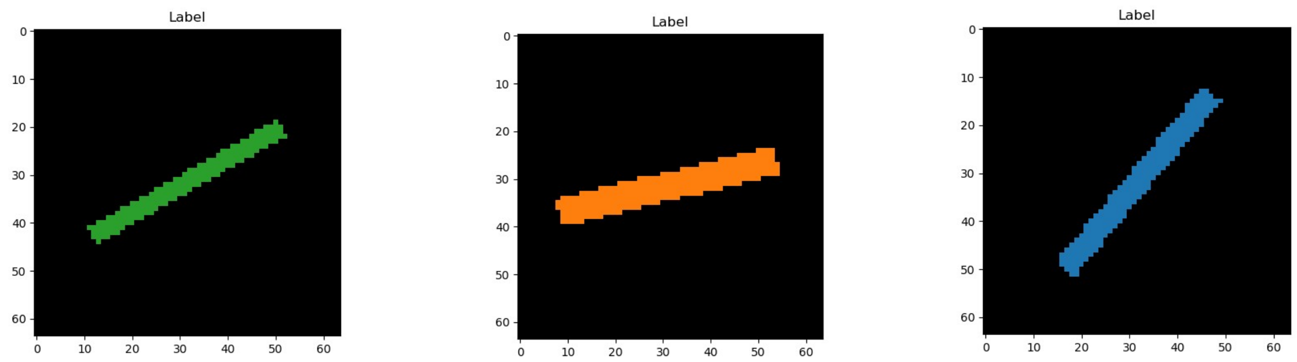
Numerical Experiments

Q-tips Dataset

Image

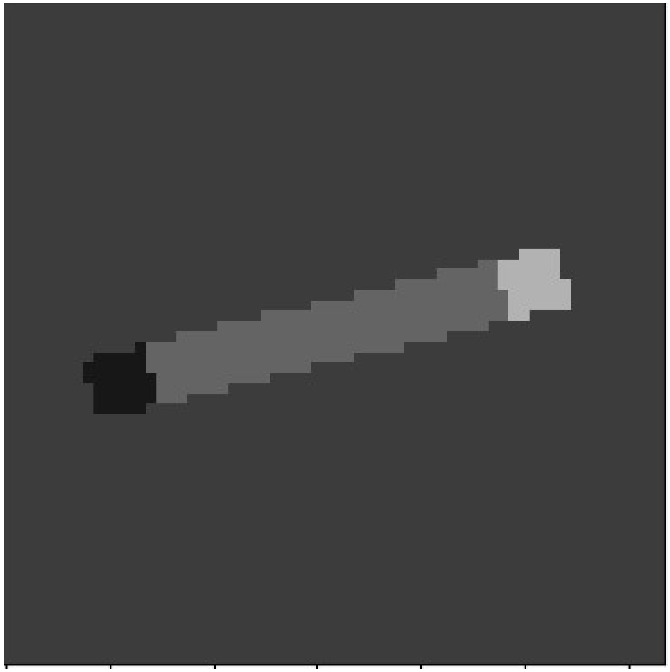


Label



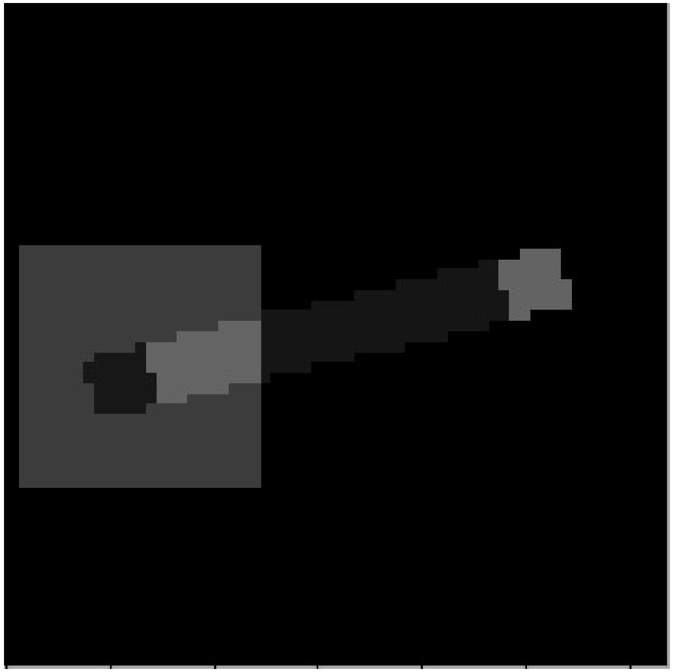
Numerical Experiments

Q-tips Dataset



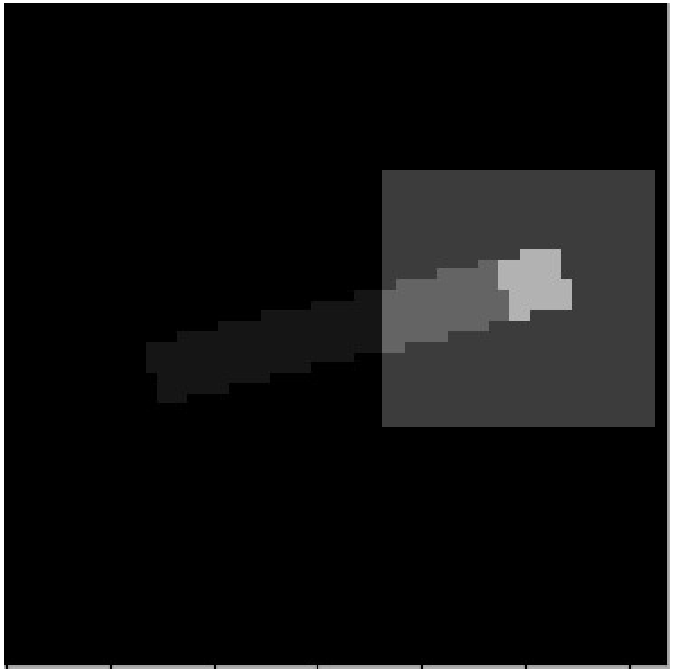
Numerical Experiments

Q-tips Dataset



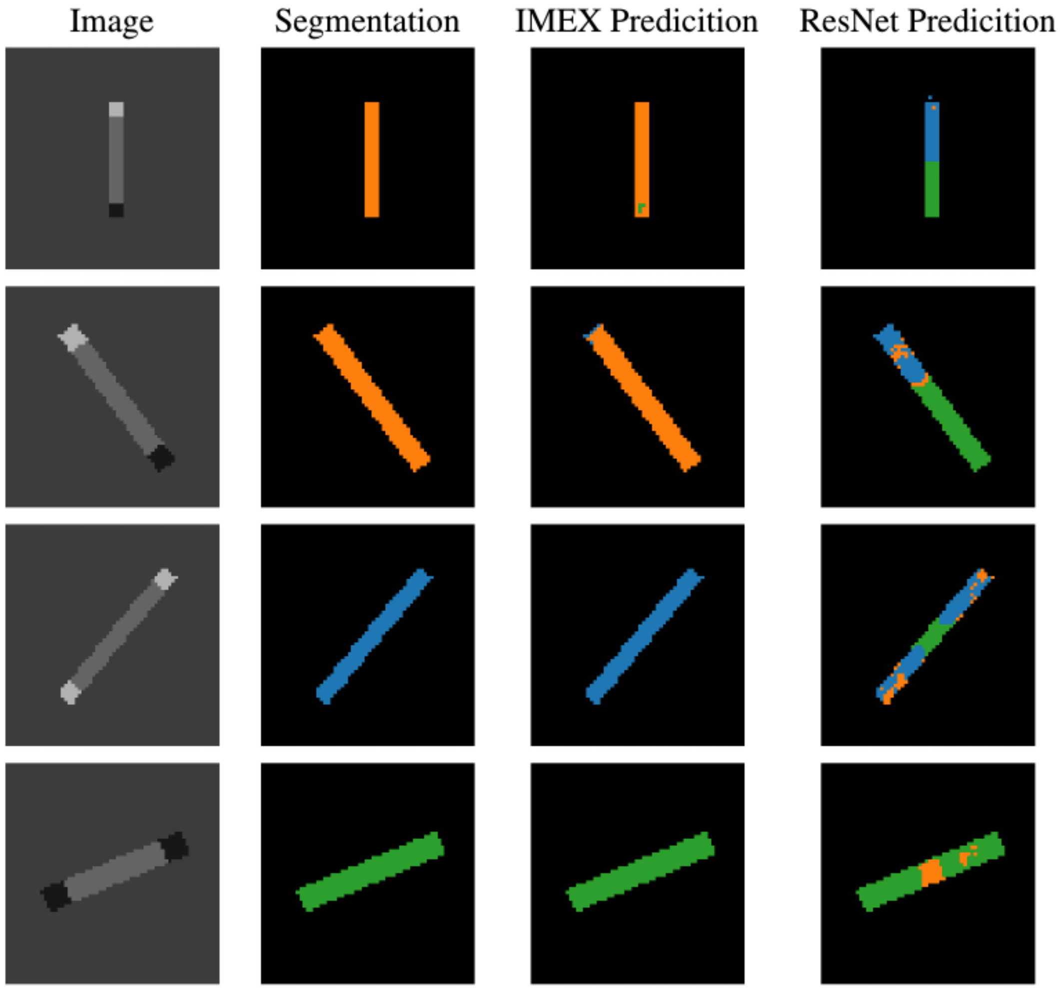
Numerical Experiments

Q-tips Dataset



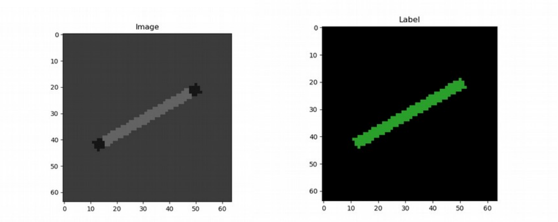
Numerical Experiments

Results - Predictions

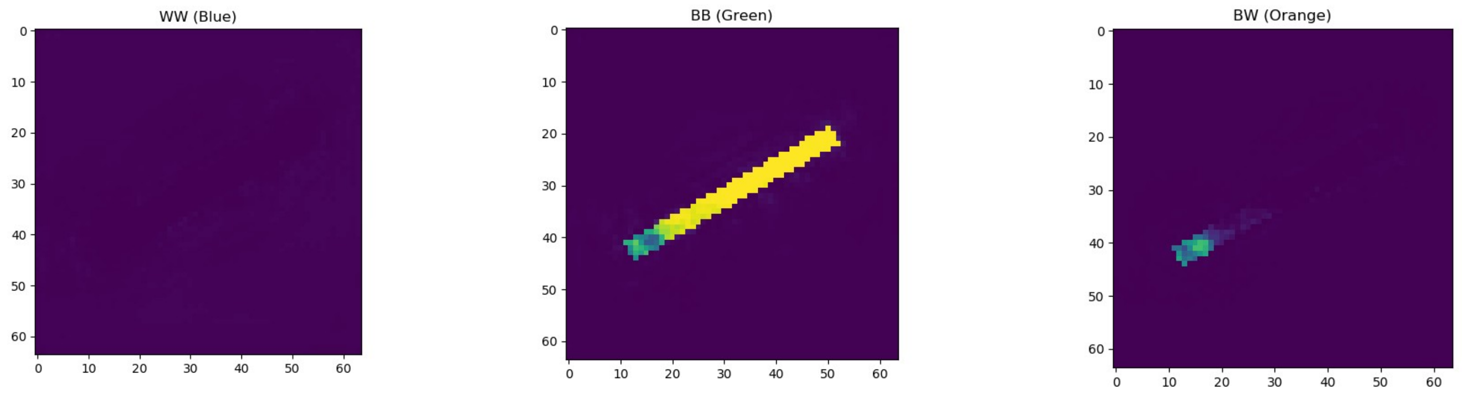


Numerical Experiments

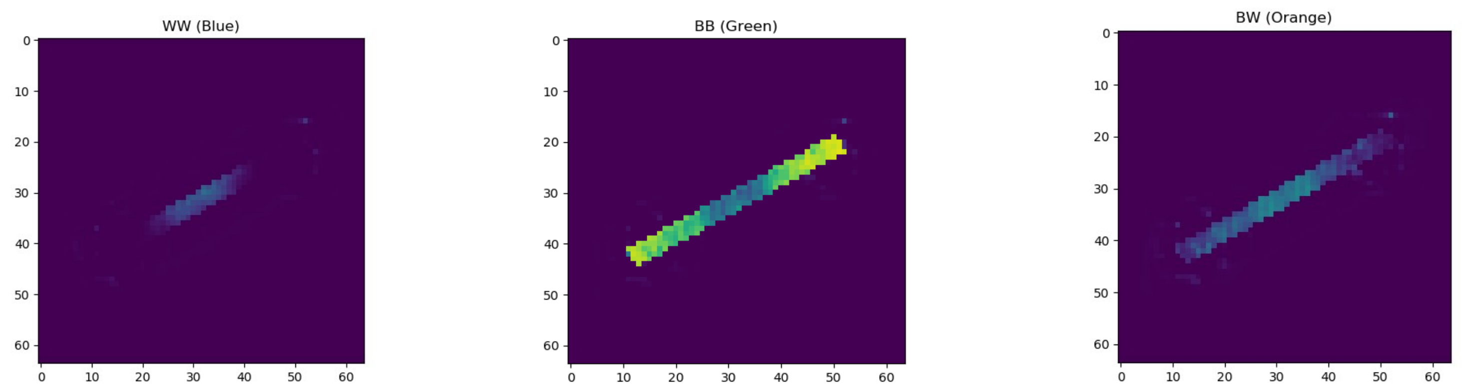
Results – Probability Maps



IMEXnet



ResNet



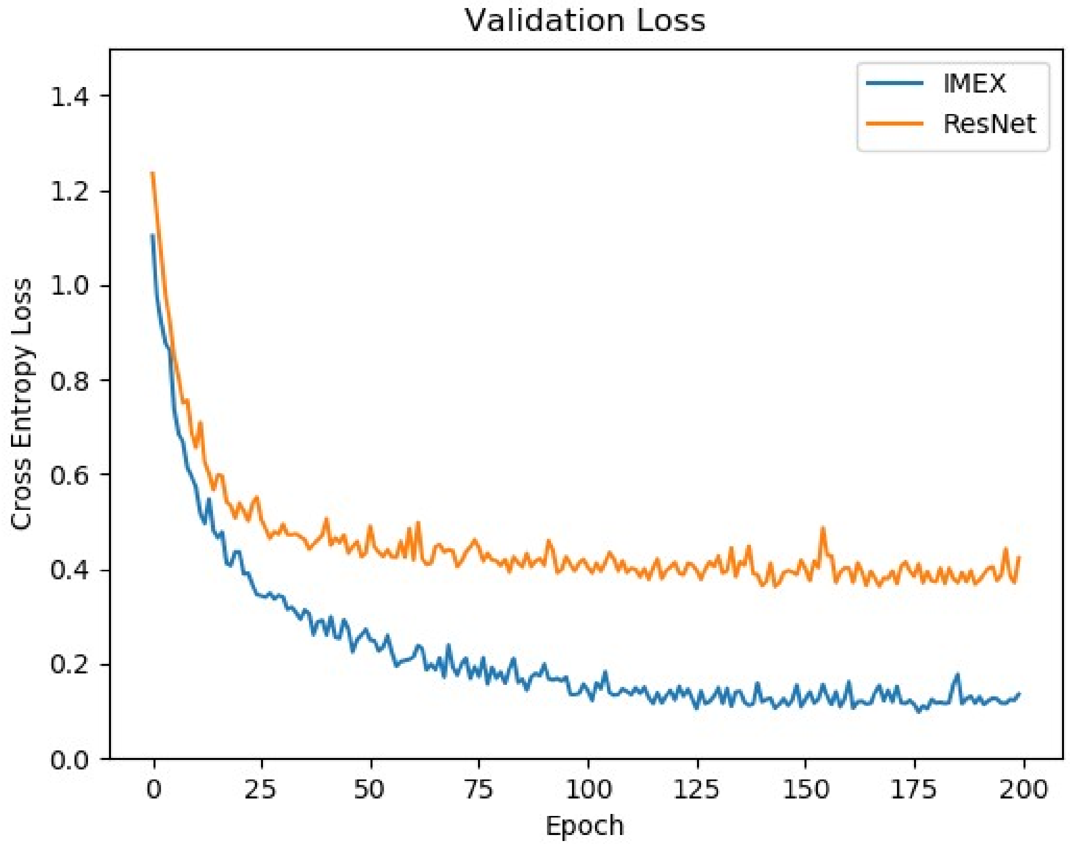
Numerical Experiments

Q-tips Results

Network	Parameters	IOU	Loss	Accuracy
IMEXnet	2701440	0.926	0.0982	99.56
ResNet	2691648	0.741	0.3332	98.18

Numerical Experiments

Q-tips Results



NYU Depth Results

Kitchen scene

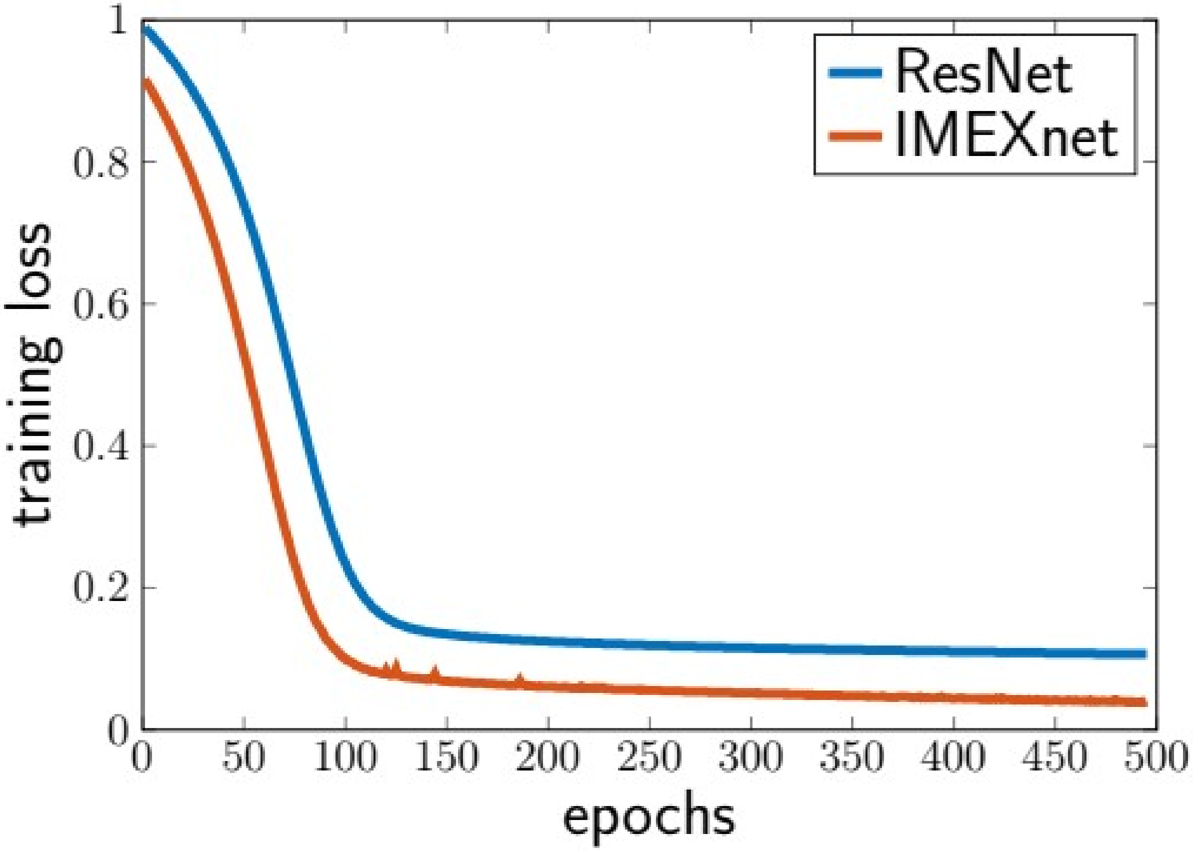
Depth map

ResNet recovery

Implicit net recovery

Numerical Experiments

NYU Depth Results



Receptive Field

How can we increase a network's receptive field?

1) Convolutions
   - Pros: Increase model complexity, increase receptive field
   - Cons: More expensive, less stable
2) Coarsening
   - Pros: Reduce dimensionality, increase receptive field
   - Cons: Lose information
3) Implicit Steps
   - Pros: Increase stability, cheap, couple all pixels
   - Cons: Additional hyperparameters, periodicity (depending on implementation)

Outline

1) **Introduction**
    1) High vs low dimensional targets
    2) Receptive Field
    3) Neural Network Stability
2) **IMEXnet**
    1) Formulation
    2) Implementation
3) **Numerical Experiments**
    1) Q-tips
    2) NYU Depth

Acknowledgments

Paper:

Haber, E., Lensink, K., Triester, E., Ruthotto, L. IMEXnet – A forward stable deep neural network. 2019

Code:

github.com/HaberGroup/SemiImplicitDNNs

Acknowledgments

This work is supported by the Mitacs Accerlerate program, Xtract AI, and the US National Science Foundation. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).