# Sparse, High Dimensional Filtering

Peter Gehler

Martin Kiefel

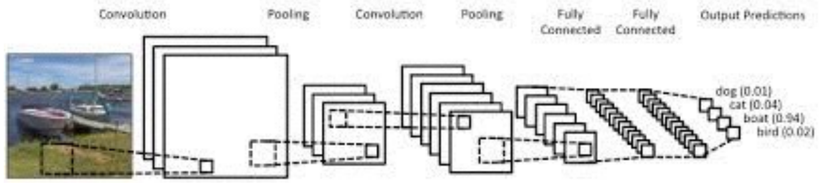
Varun Jampani

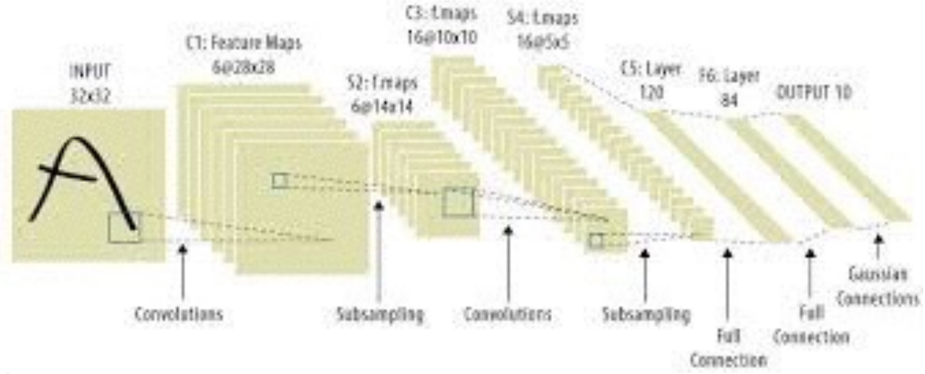
Raghudeep Gadde

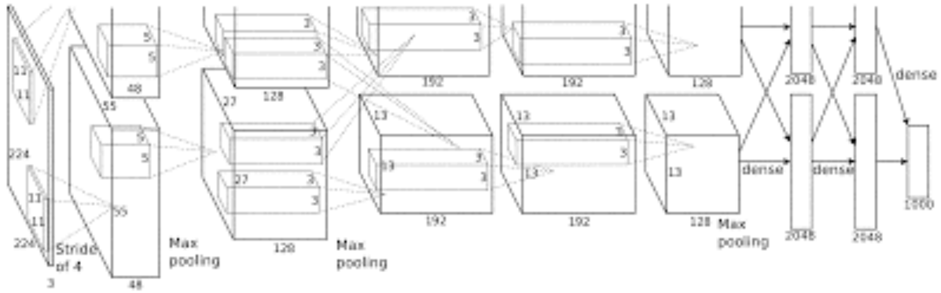
Bernstein Center for
Computational Neuroscience
Tübingen

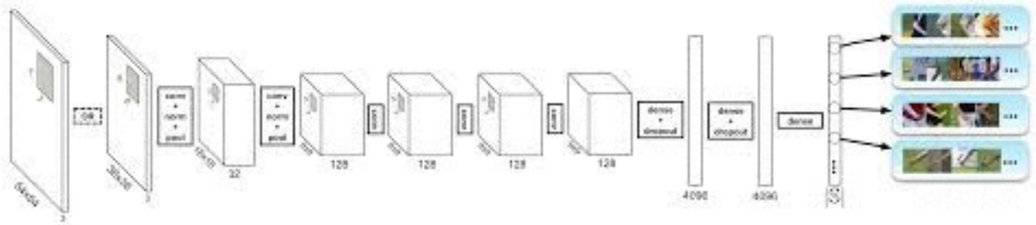
Max Planck Institute for
Intelligent Systems
TÜBINGEN

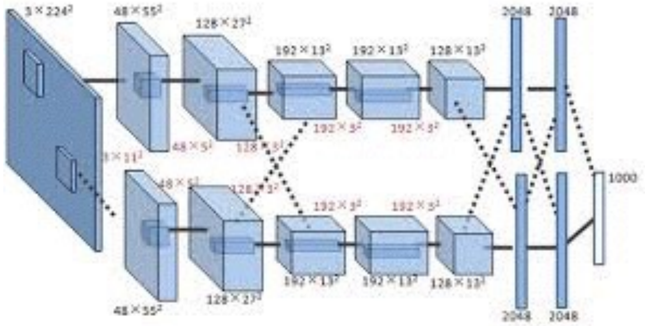
# What all CNNs have in common

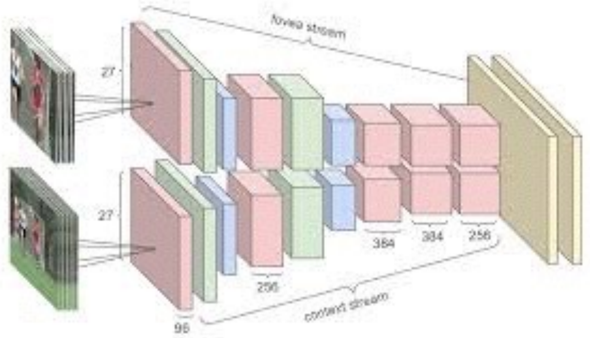


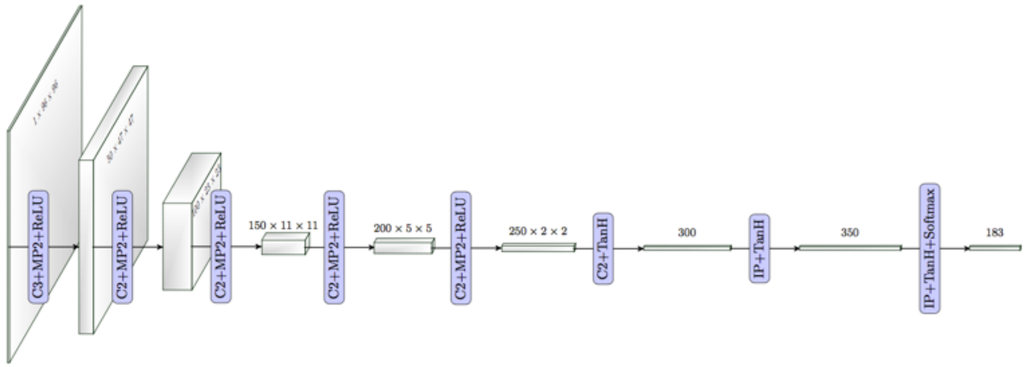



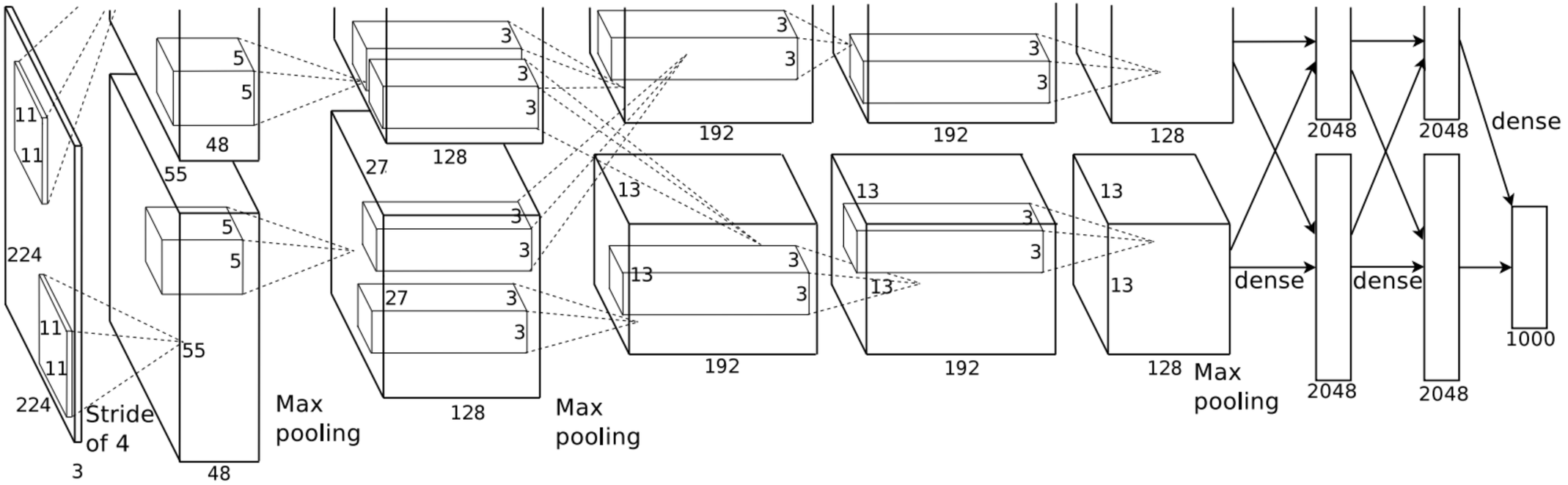





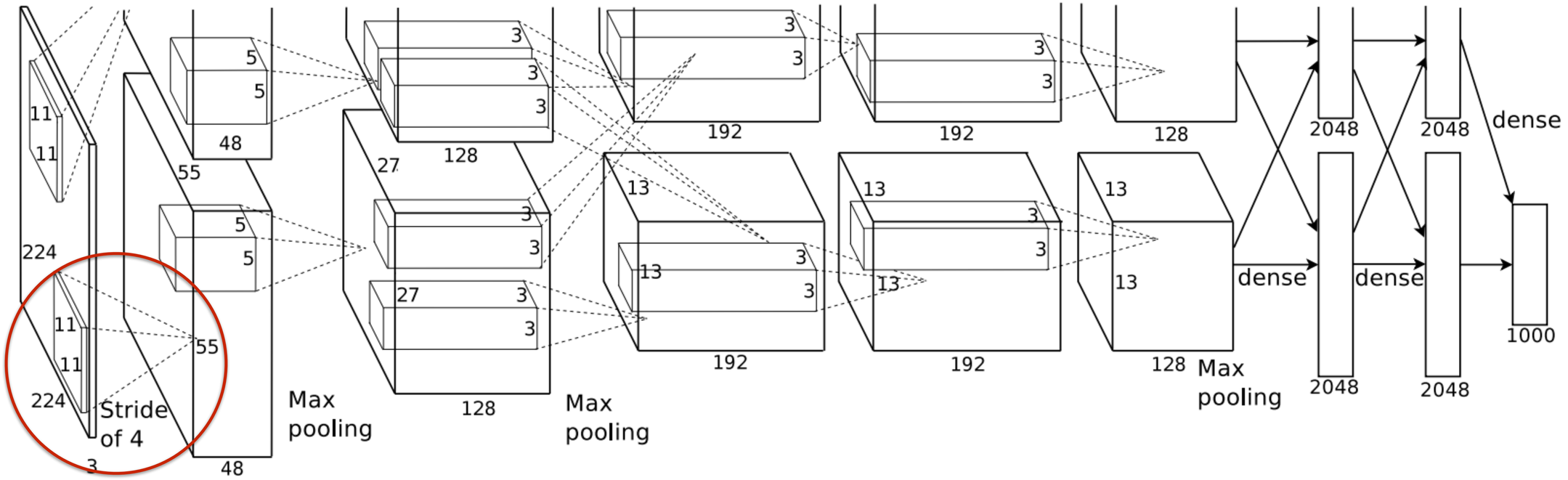



# Example: AlexNet



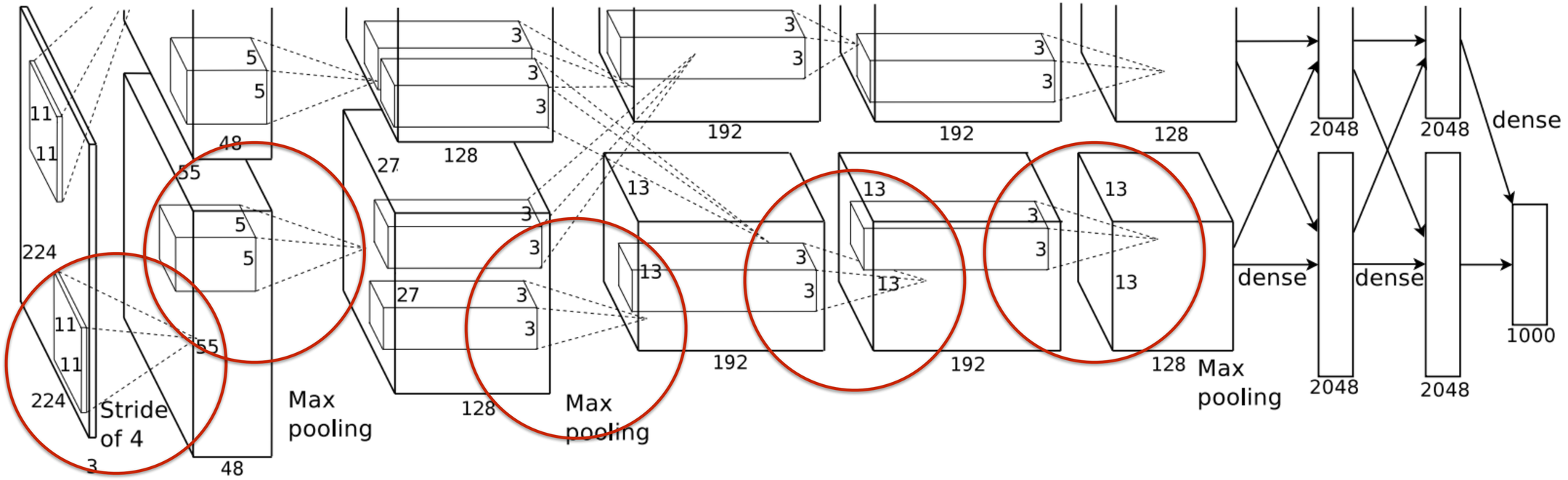

Krizhevsky, Sutskever and Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

# Example: AlexNet




Krizhevsky, Sutskever and Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

# Example: AlexNet




Krizhevsky, Sutskever and Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, NIPS 2012

1) A simple Insight

2) Consequences

1) A simple Insight

2) Consequences

# Bilateral Filtering

Input Signal

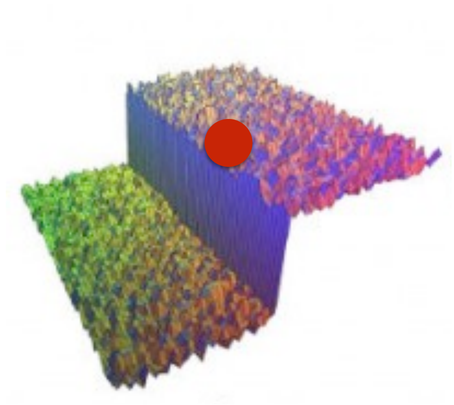


Illustration from Paris et al.

$$I(x) = \sum_{x_i \in \Omega} \mathcal{F}(\|f(x) - f(x_i)\|) I(x_i)$$

$$= \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1} (f(x) - f(x_i))\right) I(x_i)$$

Aurich and Weule, *Non-Linear Gaussian Filters Performing Edge Preserving Diffusion*, DAGM 1995
Smith and Brady, *SUSAN - A New Approach to Low Level Image Processing*, IJCV 1997
Tomasi and Manduchi, *Bilateral Filtering or Gray and Color Images*, ICCV 1998

# Bilateral Filtering

Input Signal $f(x) = \begin{pmatrix} u \\ v \end{pmatrix}$

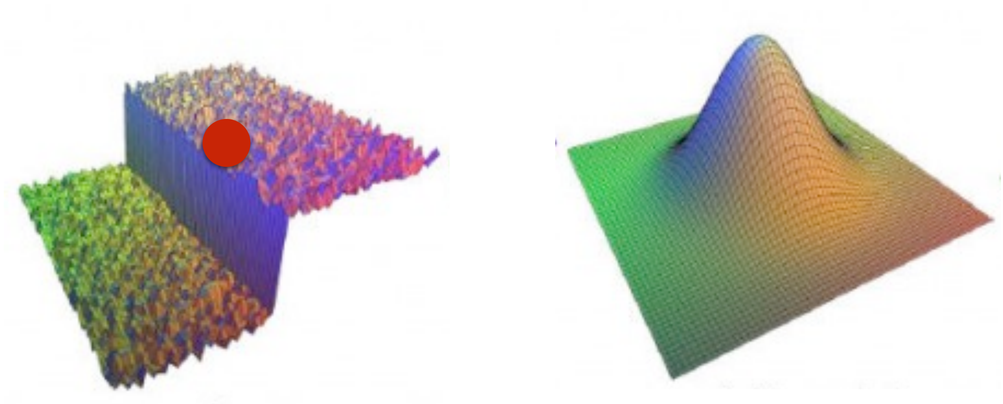


Illustration from Paris et al.

$$I(x) = \sum_{x_i \in \Omega} \mathcal{F}(\|f(x) - f(x_i)\|) I(x_i)$$

$$= \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1} (f(x) - f(x_i))\right) I(x_i)$$

Aurich and Weule, *Non-Linear Gaussian Filters Performing Edge Preserving Diffusion*, DAGM 1995
Smith and Brady, *SUSAN - A New Approach to Low Level Image Processing*, IJCV 1997
Tomasi and Manduchi, *Bilateral Filtering or Gray and Color Images*, ICCV 1998

# Bilateral Filtering

Input Signal $f(x) = \begin{pmatrix} u \\ v \end{pmatrix}$ $f(x) = \begin{pmatrix} u \\ v \\ I \end{pmatrix}$

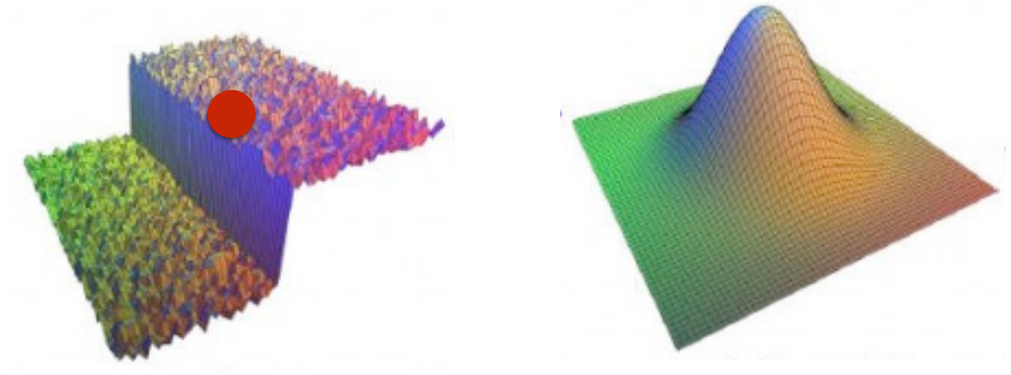


Illustration from Paris et al.

$$I(x) = \sum_{x_i \in \Omega} \mathcal{F}(\|f(x) - f(x_i)\|) I(x_i)$$

$$= \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1} (f(x) - f(x_i))\right) I(x_i)$$

Aurich and Weule, *Non-Linear Gaussian Filters Performing Edge Preserving Diffusion*, DAGM 1995
Smith and Brady, *SUSAN - A New Approach to Low Level Image Processing*, IJCV 1997
Tomasi and Manduchi, *Bilateral Filtering or Gray and Color Images*, ICCV 1998

# Bilateral Filtering

Input Signal

$$f(x) = \begin{pmatrix} u \\ v \end{pmatrix}$$

$$f(x) = \begin{pmatrix} u \\ v \\ I \end{pmatrix}$$

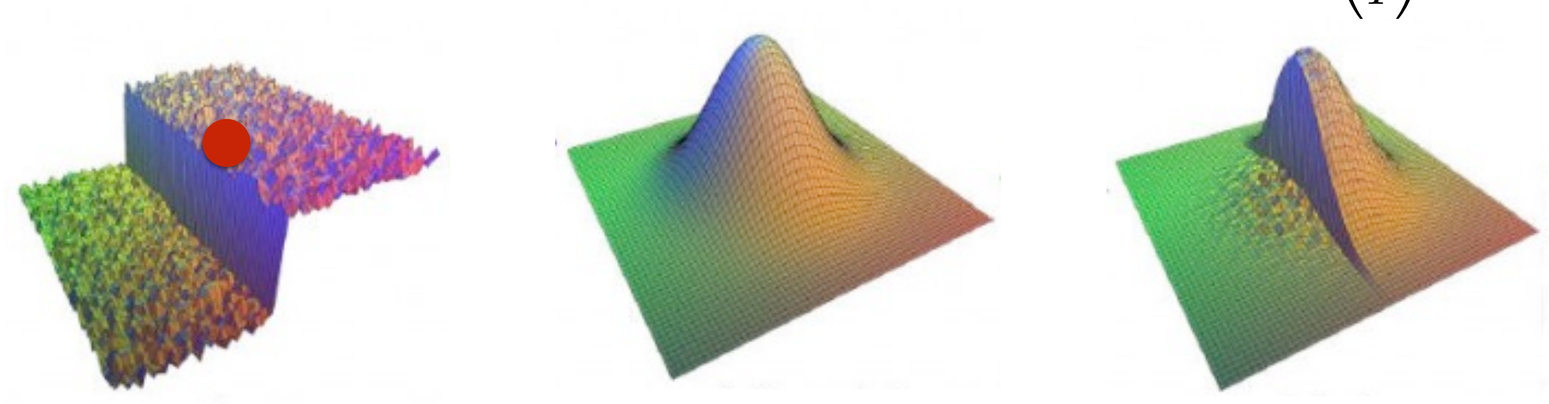


Illustration from Paris et al.

$$I(x) = \sum_{x_i \in \Omega} \mathcal{F}(\|f(x) - f(x_i)\|) I(x_i)$$

$$= \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1} (f(x) - f(x_i))\right) I(x_i)$$

Aurich and Weule, *Non-Linear Gaussian Filters Performing Edge Preserving Diffusion*, DAGM 1995
Smith and Brady, *SUSAN - A New Approach to Low Level Image Processing*, IJCV 1997
Tomasi and Manduchi, *Bilateral Filtering or Gray and Color Images*, ICCV 1998

# Bilateral Filtering

Input Signal

$$f(x) = \begin{pmatrix} u \\ v \end{pmatrix}$$

$$f(x) = \begin{pmatrix} u \\ v \\ I \end{pmatrix}$$

Result of Filtering

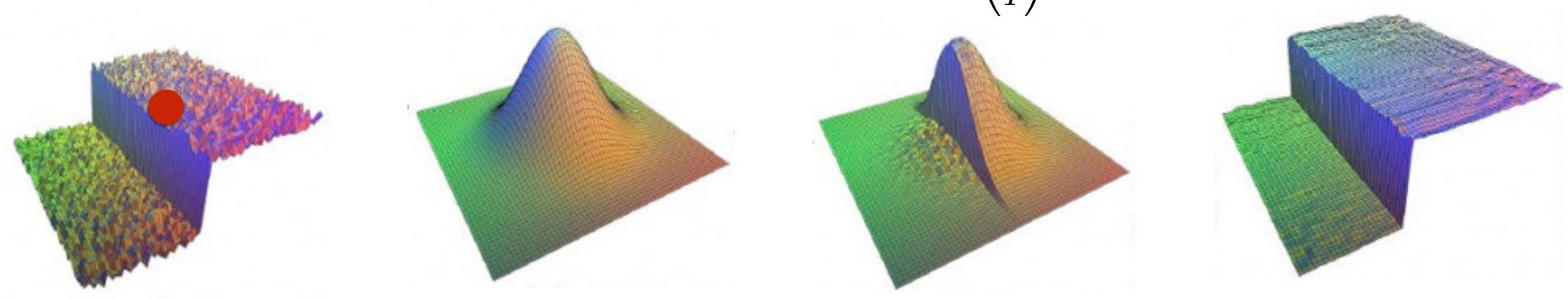


Illustration from Paris et al.

$$I(x) = \sum_{x_i \in \Omega} \mathcal{F}(\|f(x) - f(x_i)\|) I(x_i)$$

$$= \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1} (f(x) - f(x_i))\right) I(x_i)$$


Aurich and Weule, *Non-Linear Gaussian Filters Performing Edge Preserving Diffusion*, DAGM 1995

Smith and Brady, *SUSAN - A New Approach to Low Level Image Processing*, IJCV 1997

Tomasi and Manduchi, *Bilateral Filtering or Gray and Color Images*, ICCV 1998

# Bilateral Filter Preserve Edges

$$I(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} \exp\left(-\frac{1}{2}(f(x) - f(x_i))^\top \Sigma^{-1}(f(x) - f(x_i))\right) I(x_i)$$

# In Practice — Gaussian

The bilateral filter is defined as

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

where the normalization term

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

ensures that the filter preserves image energy and

- $I^{\text{filtered}}$ is the filtered image;
- $I$ is the original input image to be filtered;
- $x$ are the coordinates of the current pixel to be filtered;
- $\Omega$ is the window centered in $x$;
- $f_r$ is the range kernel for smoothing differences in intensities. This function can be a Gaussian function;
- $g_s$ is the spatial kernel for smoothing differences in coordinates. This function can be a Gaussian function;

from wikipedia
**https://en.wikipedia.org/wiki/Bilateral_filter**

# In Practice — Gaussian

The bilateral filter is defined as

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|),$$

where the normalization term

$$W_p = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$

ensures that the filter preserves image energy and

- $I^{\text{filtered}}$ is the filtered image;
- $I$ is the original input image to be filtered;
- $x$ are the coordinates of the current pixel to be filtered;
- $\Omega$ is the window centered in $x$;
- $f_r$ is the range kernel for smoothing differences in intensities. This function can be a Gaussian function;
- $g_s$ is the spatial kernel for smoothing differences in coordinates. This function can be a Gaussian function;

from wikipedia
**https://en.wikipedia.org/wiki/Bilateral_filter**

# Implementation as High Dimensional Filtering

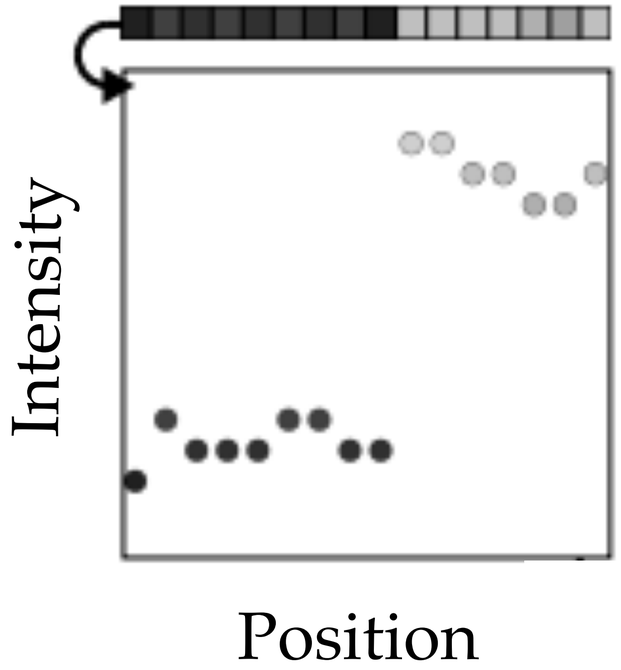



Illustration from Adams et al.

# Implementation as High Dimensional Filtering

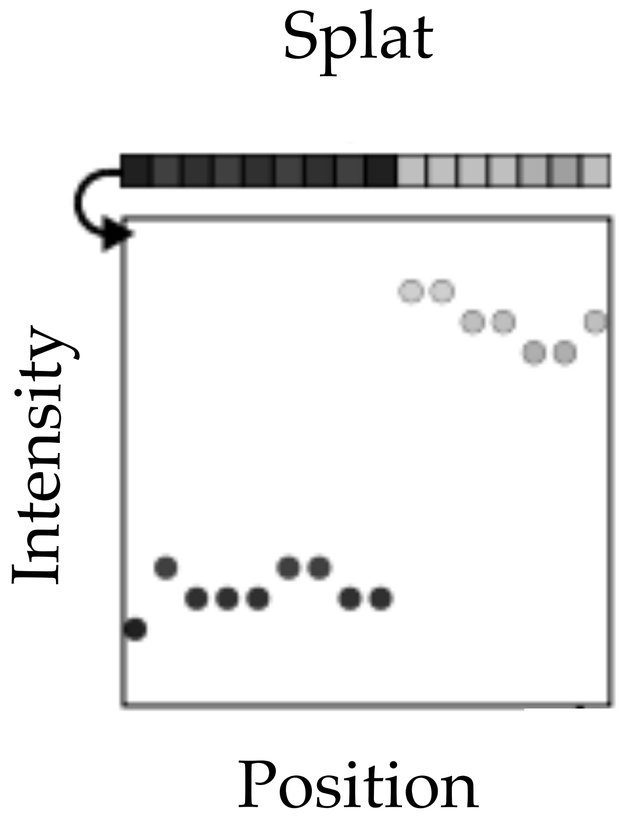



Illustration from Adams et al.

# Implementation as High Dimensional Filtering

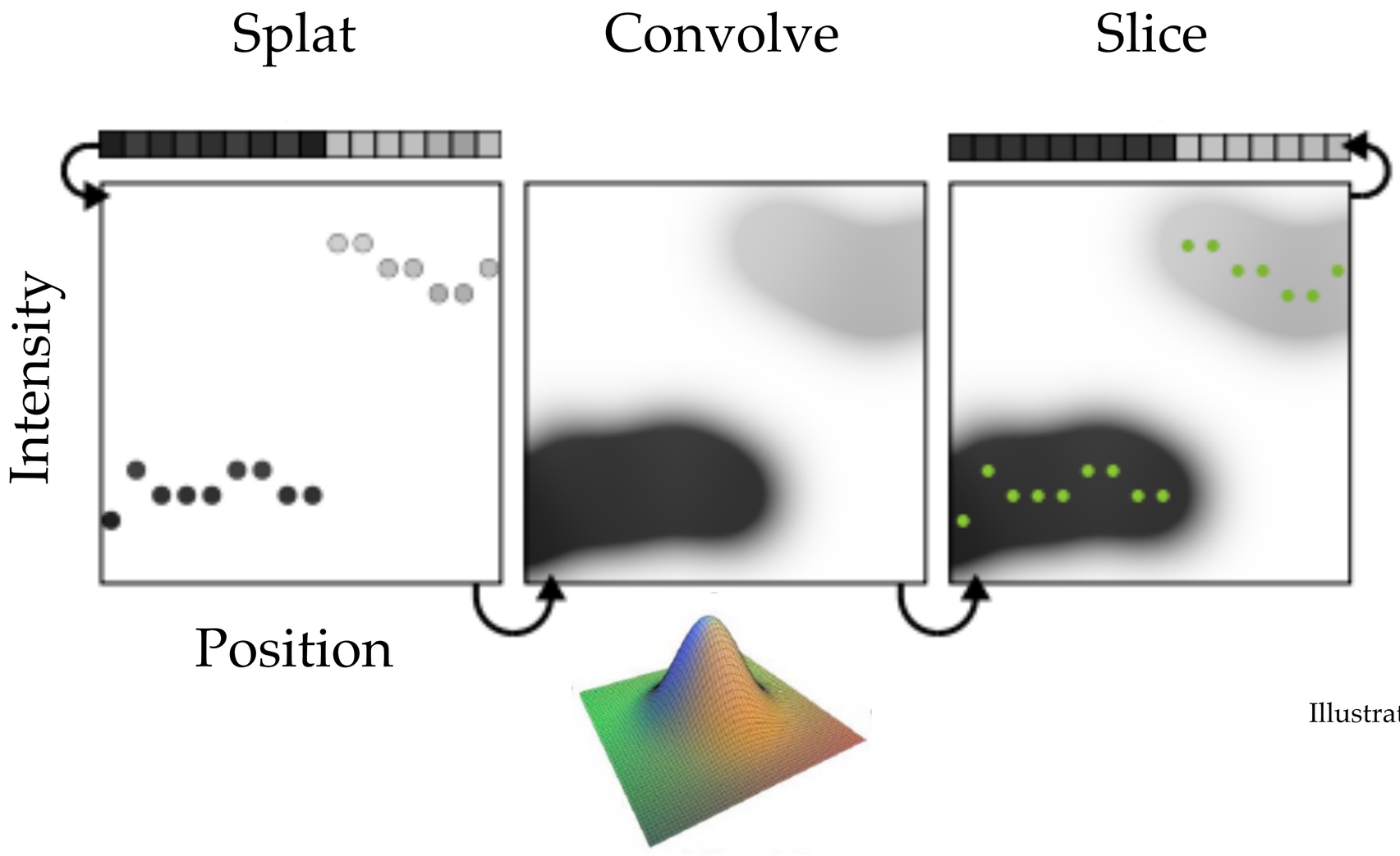
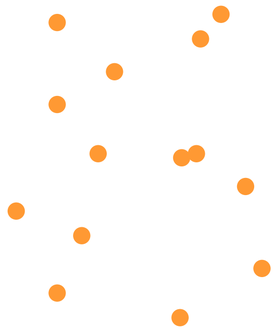



Illustration from Adams et al.

# Permutohedral Lattice

## Splat

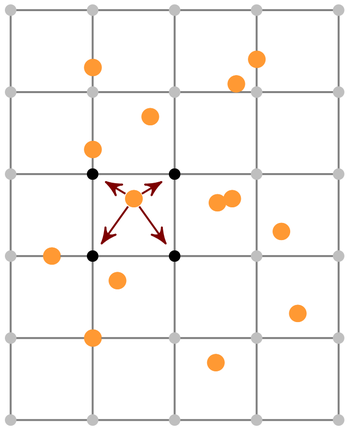
$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



# Permutohedral Lattice

Splat

$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$

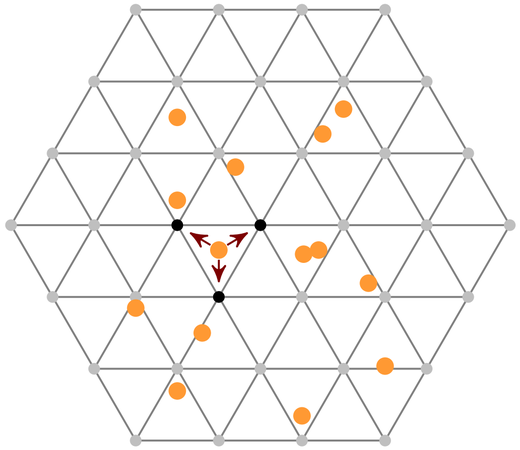


- Cell size grows exponential with dimension

# Permutohedral Lattice

Splat

$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



- Cell size grows linearly with dimension
- Quick look up and hashing used to store populated cells

Adams et al. *Fast High-Dimensional Filtering Using the Permutohedral Lattice*, Eurographics 2009

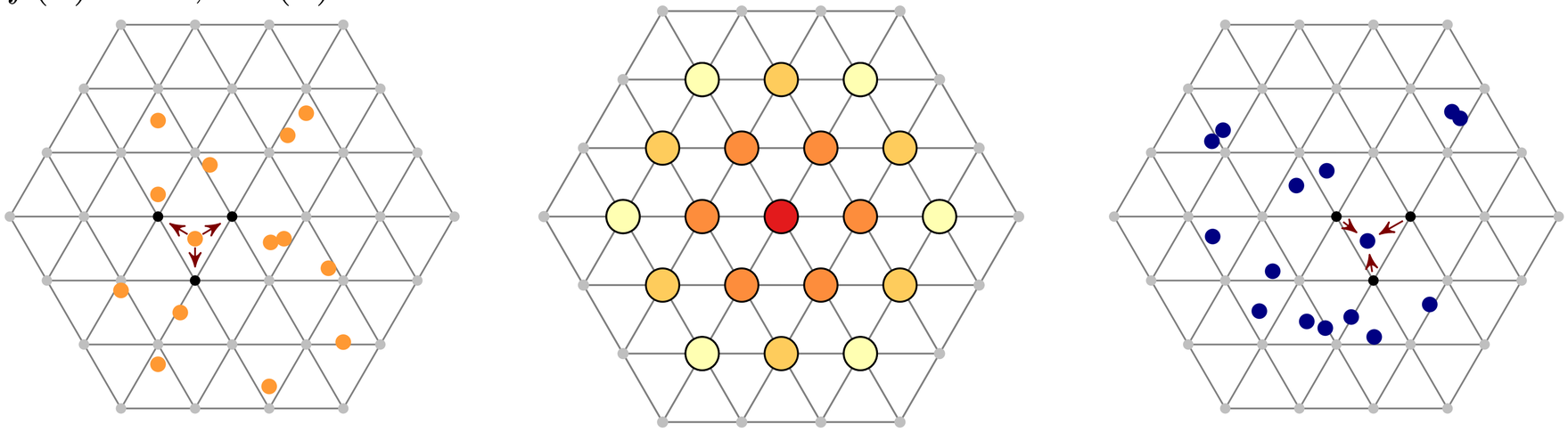
# Permutohedral Lattice

Splat

Gaussian Convolution

Slice

$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



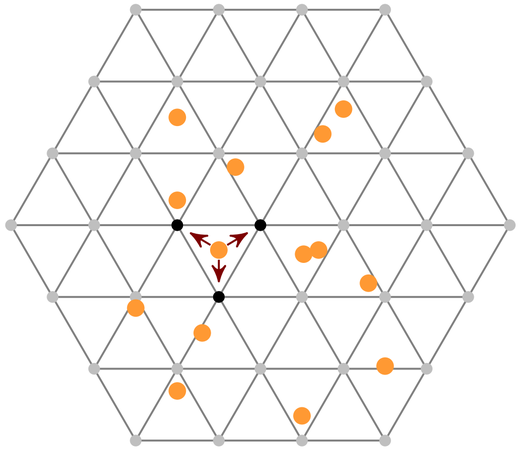
- Approximation to the Gaussian Convolution

Adams et al. *Fast High-Dimensional Filtering Using the Permutohedral Lattice*, Eurographics 2009

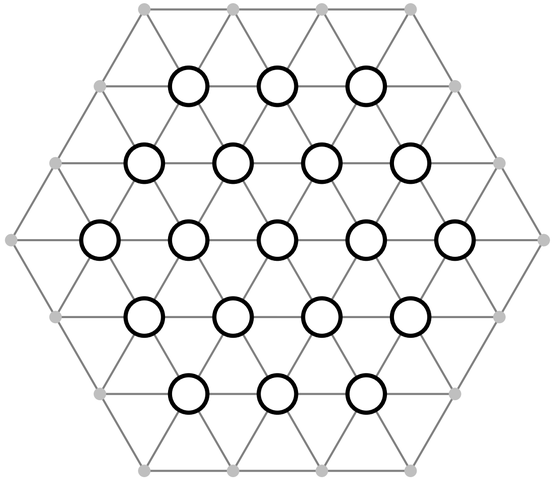
# Permutohedral Lattice

Splat

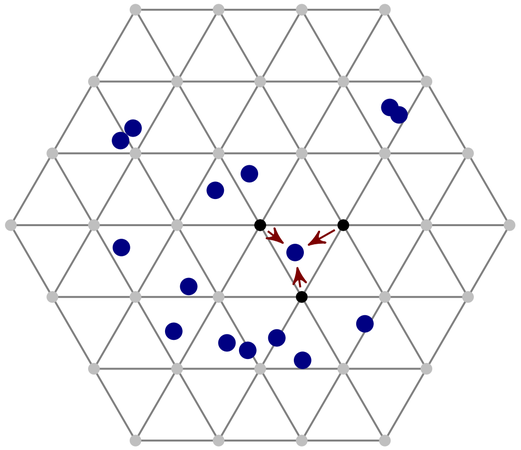
$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



Any Convolution



Slice

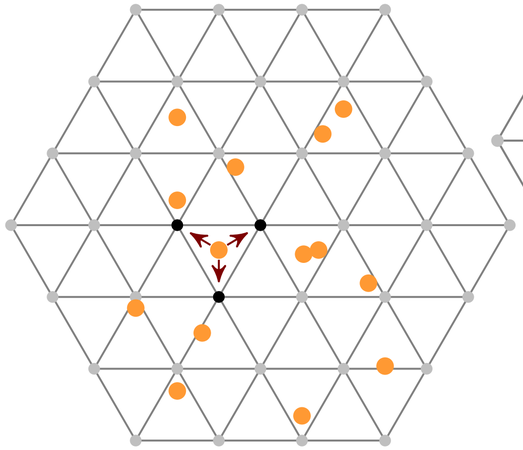


- All operations are linear — gradients simple matrix calculus
- Use any gradient based solver to learn the filter

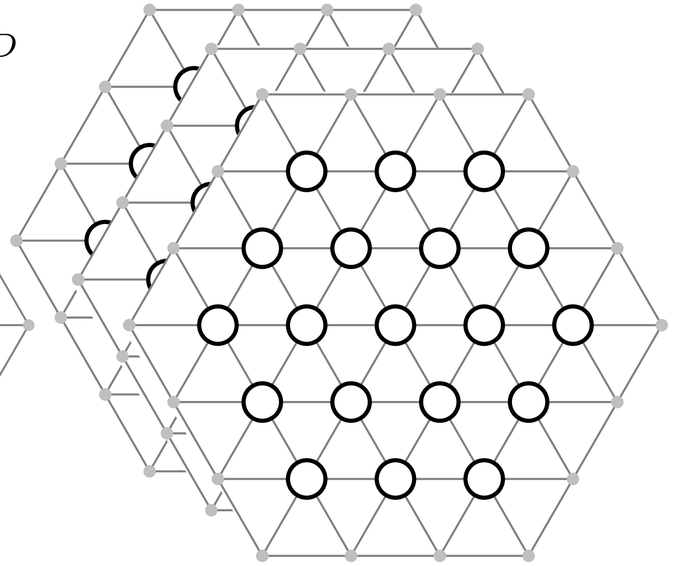
# Permutohedral CNN

Splat

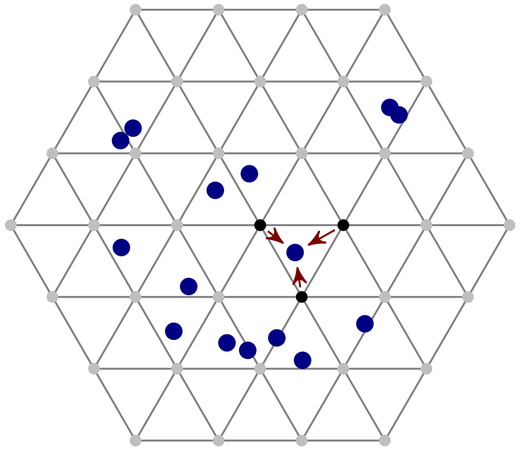
$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



Filter Bank



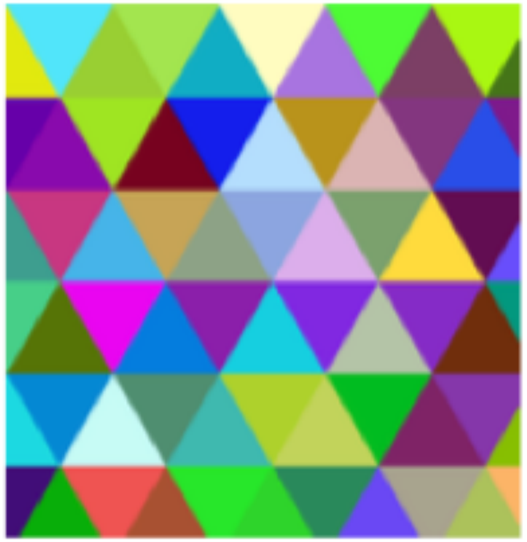
Slice



- High Dimensional and Sparse Convolution for CNNs

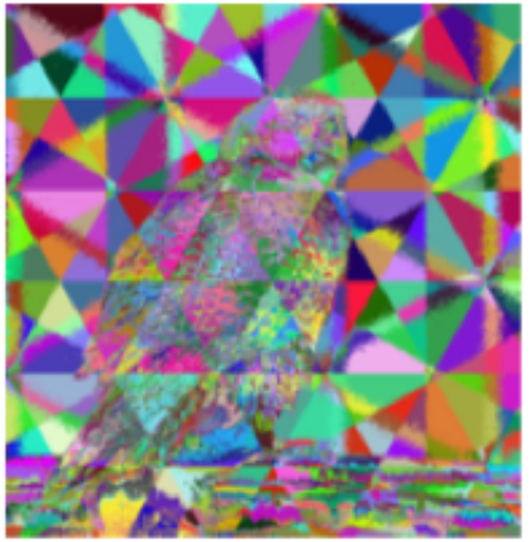
# Permuothedral Lattice Visualization

Image



Spatial Position

RGB Color



Position and Color

- Pixels in same lattice cell have the same color

# Summary: Sparse, High Dimensional Convolution

- No need for spatial grid organization, list of unordered points

$$(f_i, z_i), f_i \in \mathbb{R}^D, z_i \in \mathbb{R}^C$$

# Summary: Sparse, High Dimensional Convolution

- No need for spatial grid organization, list of unordered points

$$(f_i, z_i), f_i \in \mathbb{R}^D, z_i \in \mathbb{R}^C$$

- Filtering result

$$\hat{z}_i = \sum_j \mathcal{F}\left(\|f_i - f_j\|\right) z_j$$

# Summary: Sparse, High Dimensional Convolution

- No need for spatial grid organization, list of unordered points

$$(f_i, z_i), f_i \in \mathbb{R}^D, z_i \in \mathbb{R}^C$$

- Filtering result

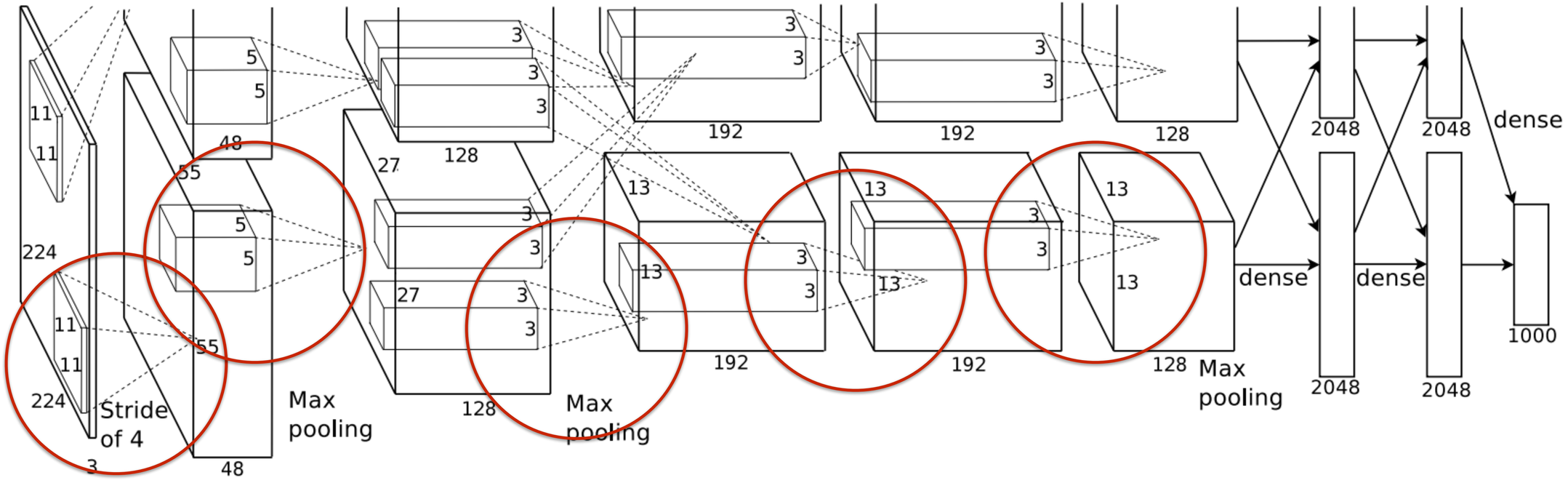
$$\hat{z}_i = \sum_j \mathcal{F}\left(\|f_i - f_j\|\right) z_j$$

- In other words: computing matrix-vector product is fast not only for Toeplitz matrices

$$\hat{Z} = A(f_{in}, f_{out})Z$$

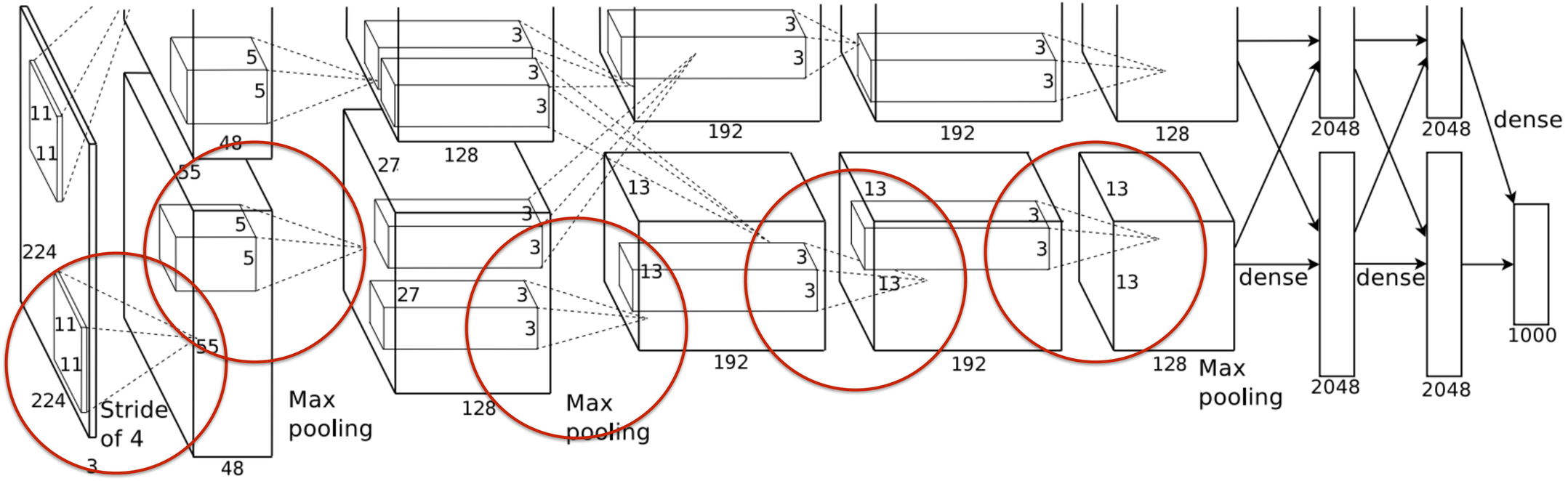
# Convolutional Neural Networks (CNN)

- Allows high-dimensional CNNs in contrast to common spatial CNNs (at slightly higher computation cost)



# Convolutional Neural Networks (CNN)

- Allows high-dimensional CNNs in contrast to common spatial CNNs (at slightly higher computation cost)
- Current research: How to organize intermediate CNN representations in higher dimensional spaces? And what are those dimensions?



# 1) A simple Insight

# 2) Consequences

Image Filtering

Bilateral CNNs

DenseCRFs

# 1) A simple Insight

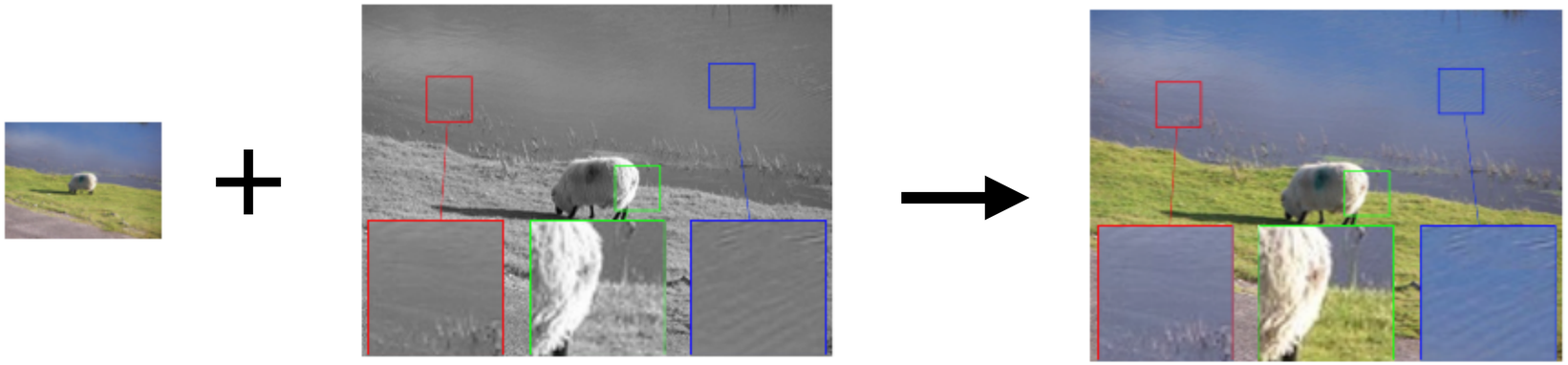
## 2) Consequences

Image Filtering

Bilateral CNNs

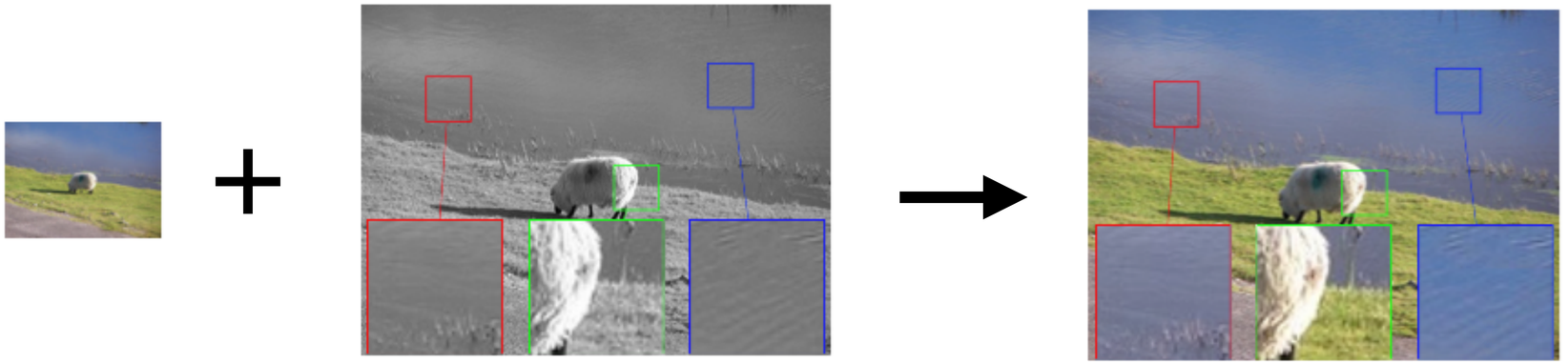
DenseCRFs

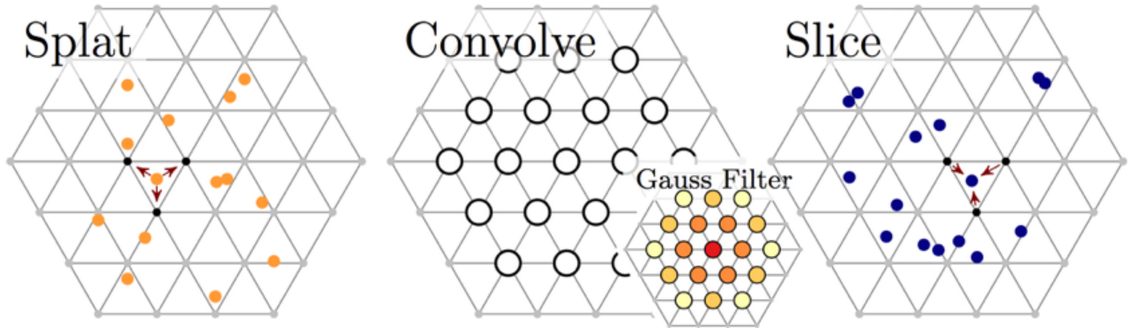
# Guided Upsampling



Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

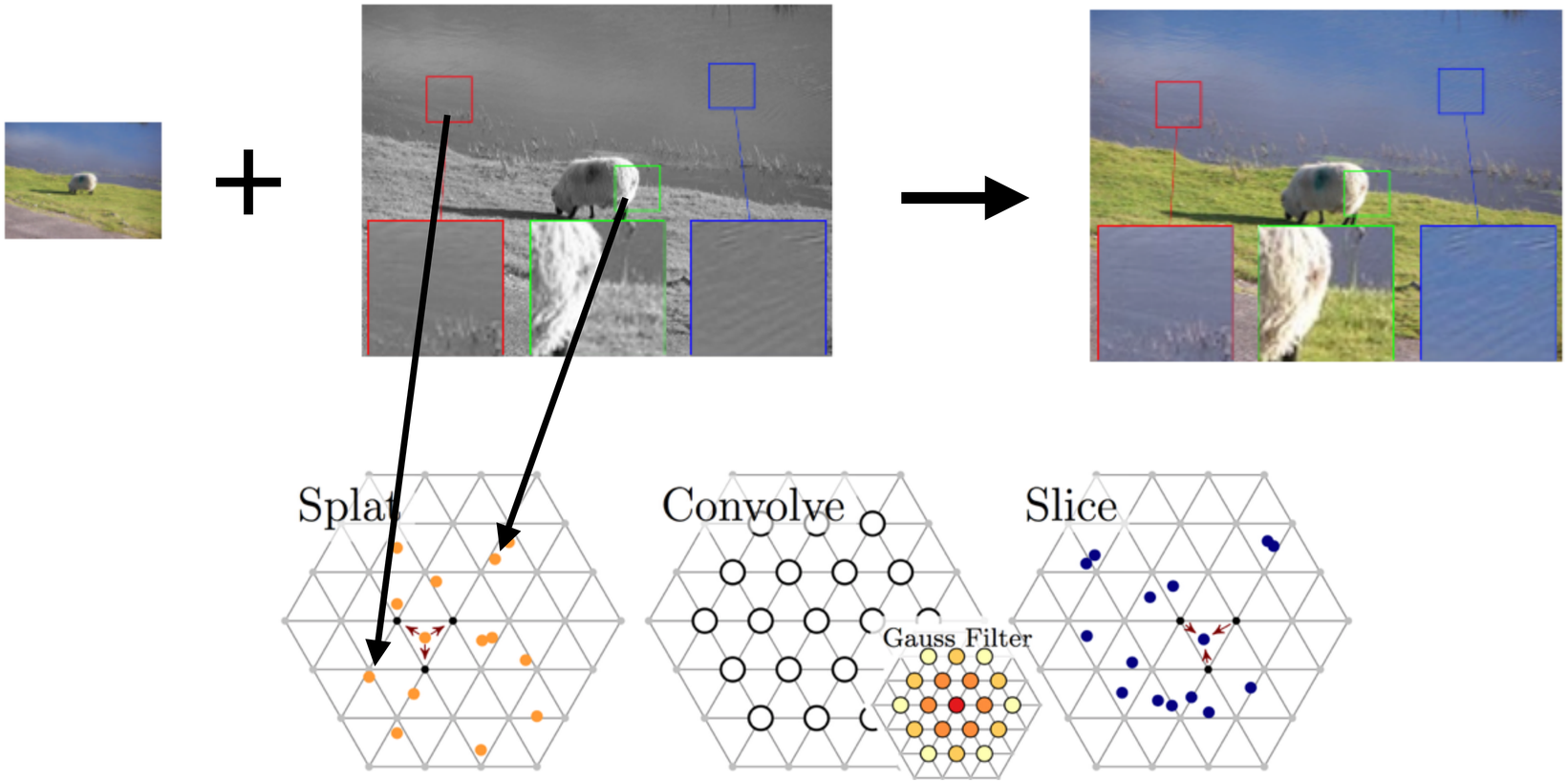
# Guided Upsampling





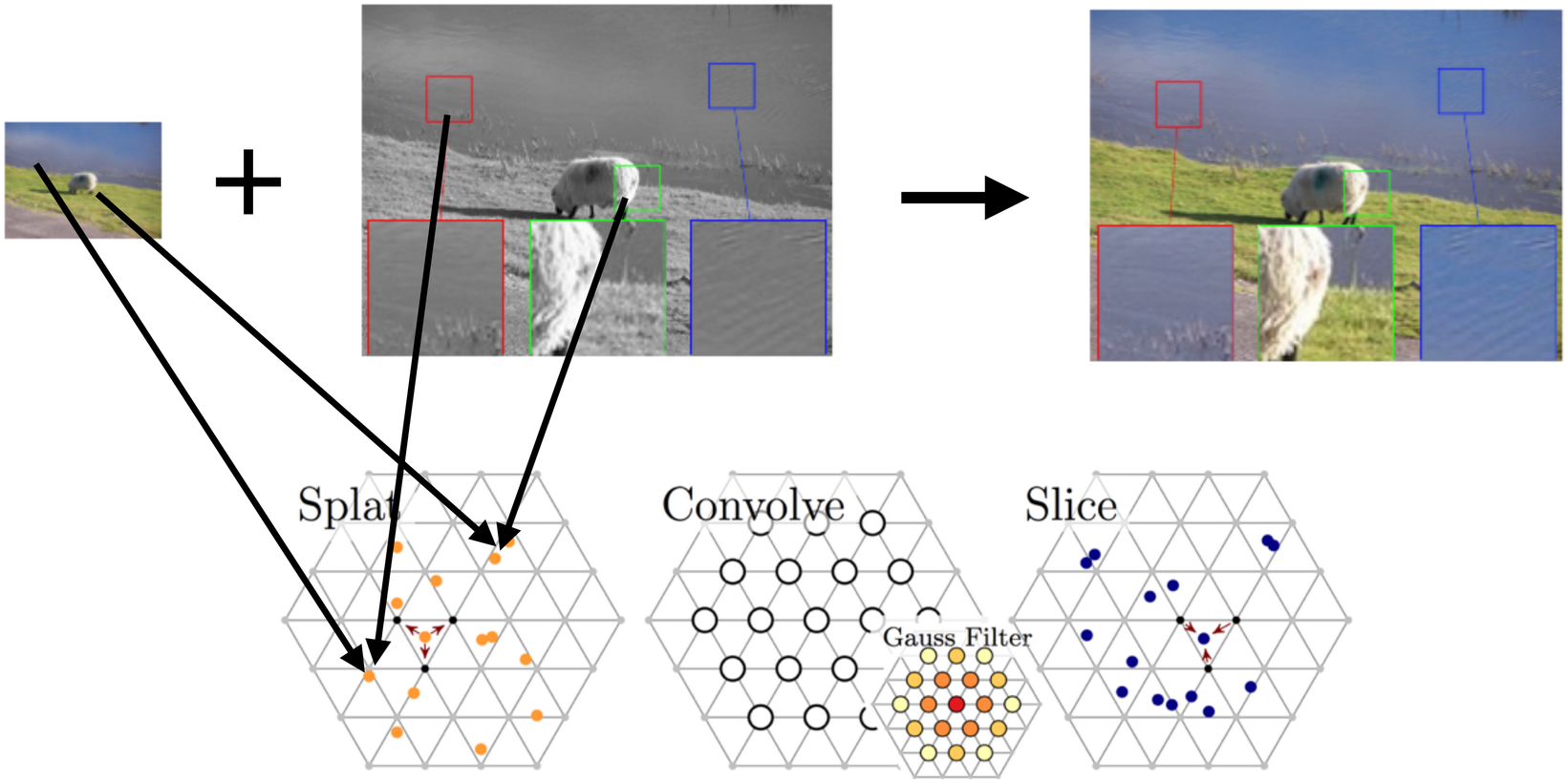

Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

# Guided Upsampling



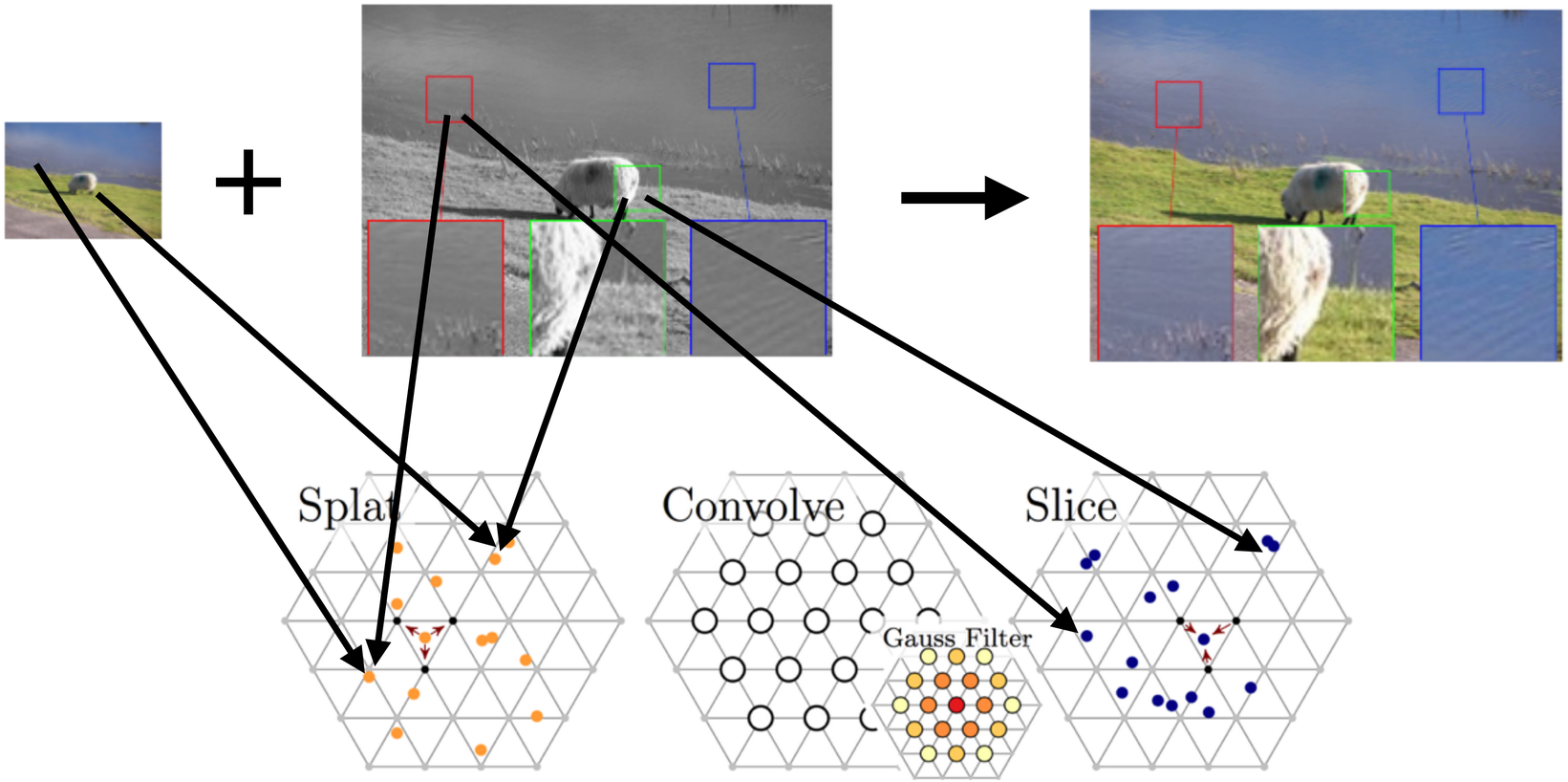

Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

# Guided Upsampling



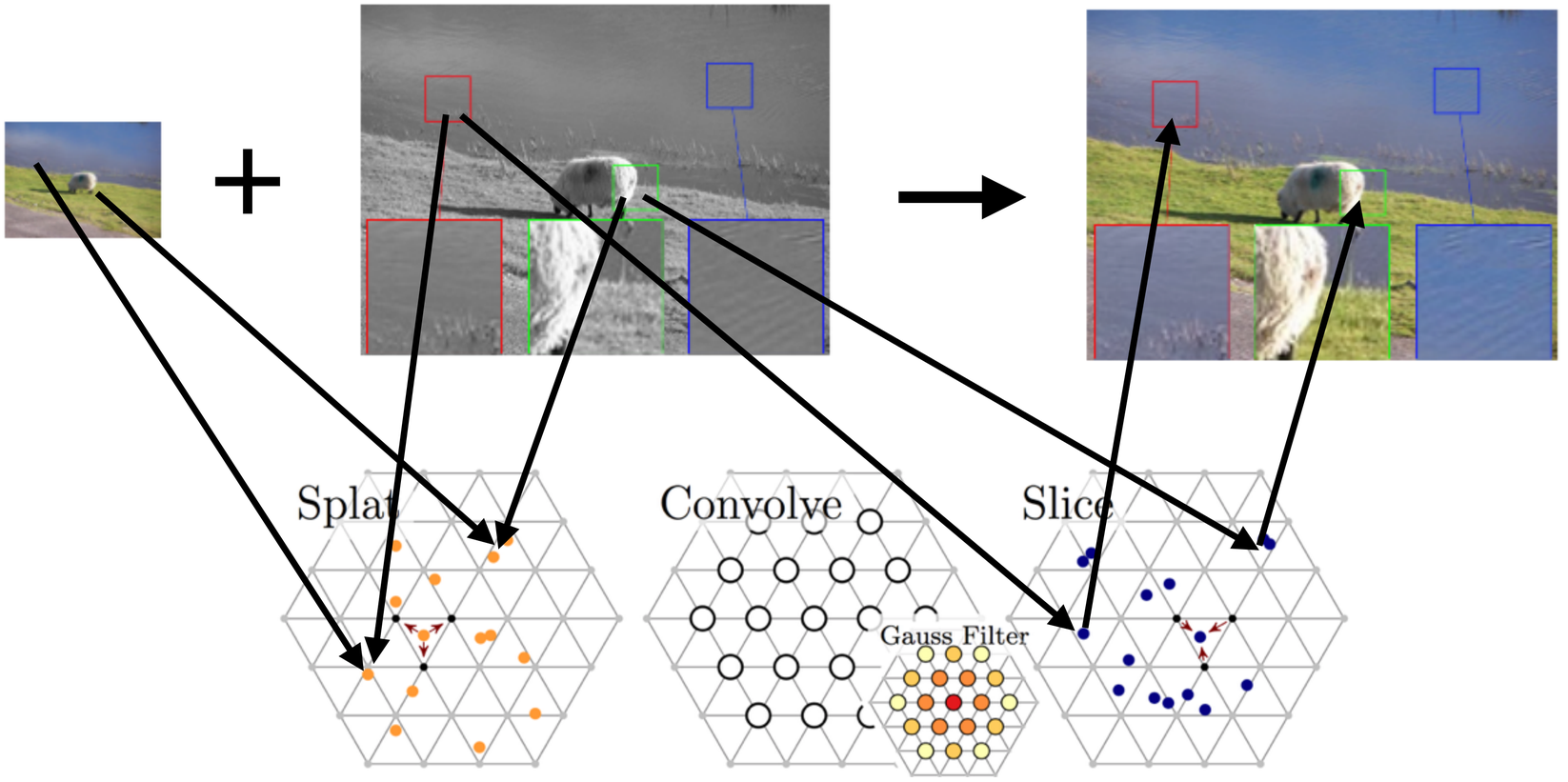

Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

# Guided Upsampling



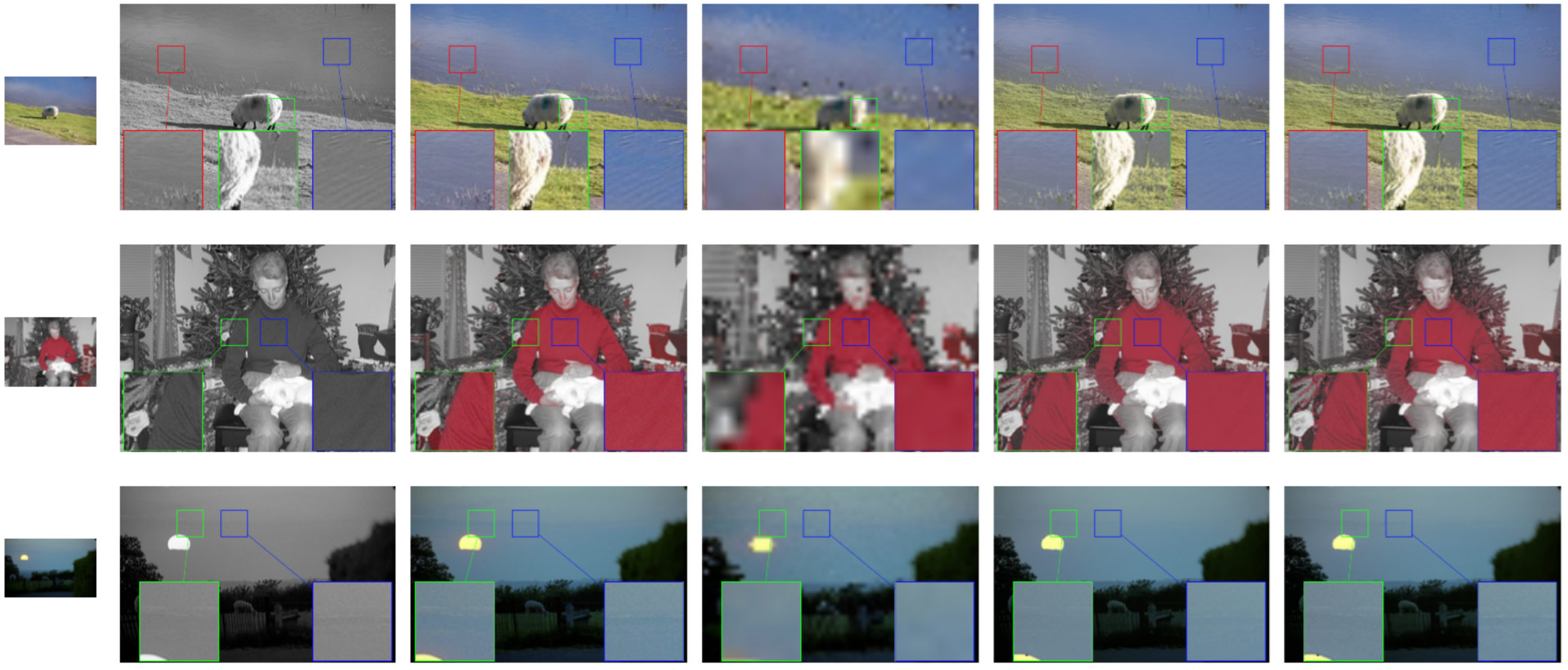

Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

# Guided Upsampling



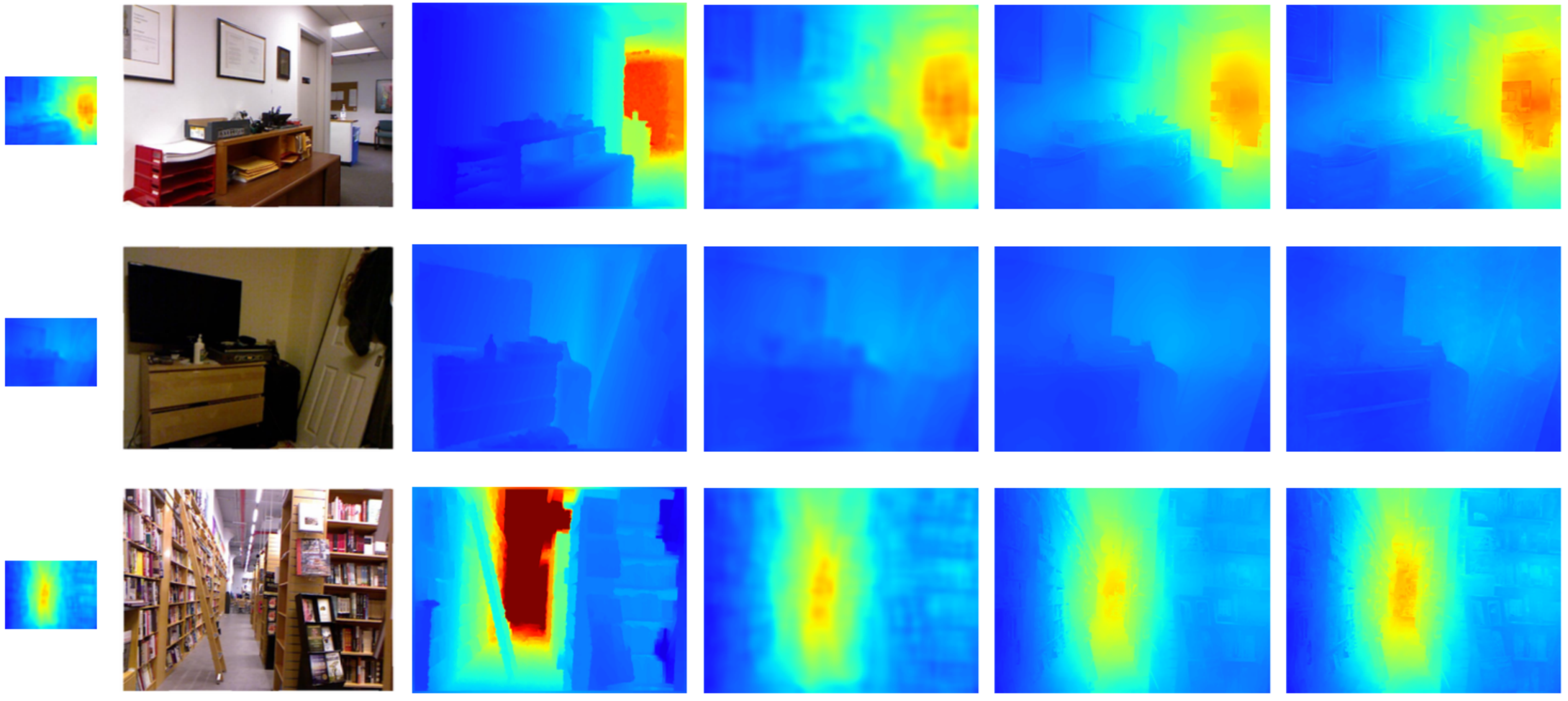

Kopf, Cohen, Lischinski and Uyttendaele, **Joint Bilateral Upsampling**, SIGGRAPH, 2007

# Color Upsampling (8x)



(a) Input (b) Gray Guidance (c) Ground Truth (d) Bicubic Interpolation (e) Gauss Bilateral (f) Learned Bilateral

# Depth Upsampling (8x)



(a) Input (b) Guidance (c) Ground Truth (d) Bicubic Interpolation (e) Gauss Bilateral (f) Learned Bilateral

# Joint Bilateral Upsampling

- Train/Test on Pascal VOC12 segmentation
  - PSNR (higher better)
- Train / Test on NYU Depth
  - RMSE (lower better)

| Upsampling factor | Bicubic | Gaussian | Learned |
|---|---|---|---|
| Color Upsampling (PSNR) | | | |
| **2x** | 24.19 | 33.46 | **34.05** |
| **4x** | 20.34 | 31.87 | **32.28** |
| **8x** | 17.99 | 30.51 | **30.81** |
| **16x** | 16.10 | 29.19 | **29.52** |
| Depth Upsampling (RMSE) | | | |
| **8x** | 0.753 | 0.753 | **0.748** |

# 1) A simple Insight

# 2) Consequences

Image Filtering

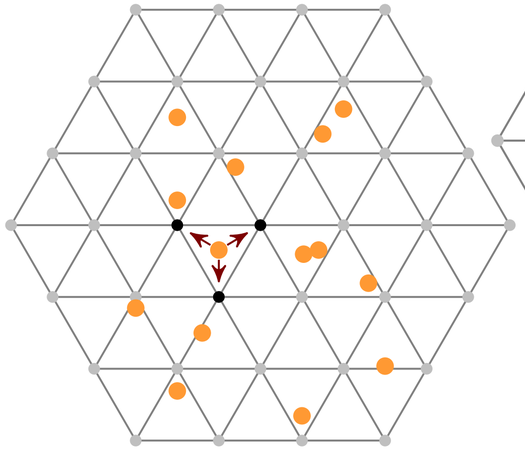
Bilateral CNNs

DenseCRFs

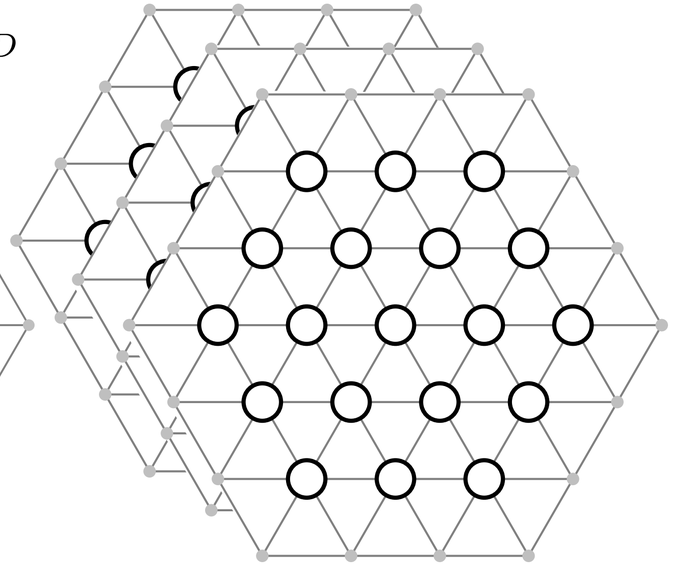
# Bilateral CNNs

Splat

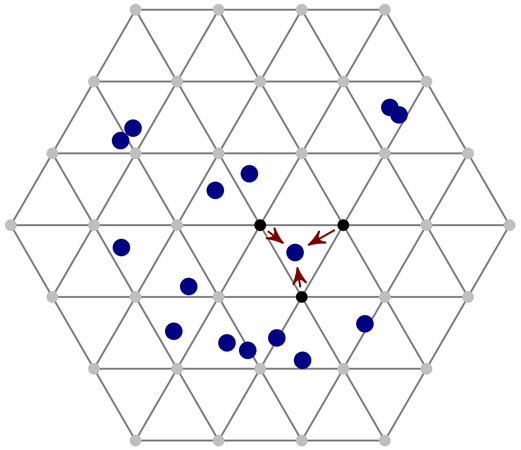
$f(x) \in \mathbb{R}^2, \quad I(x) \in \mathbb{R}^D$



Filter Bank



Slice

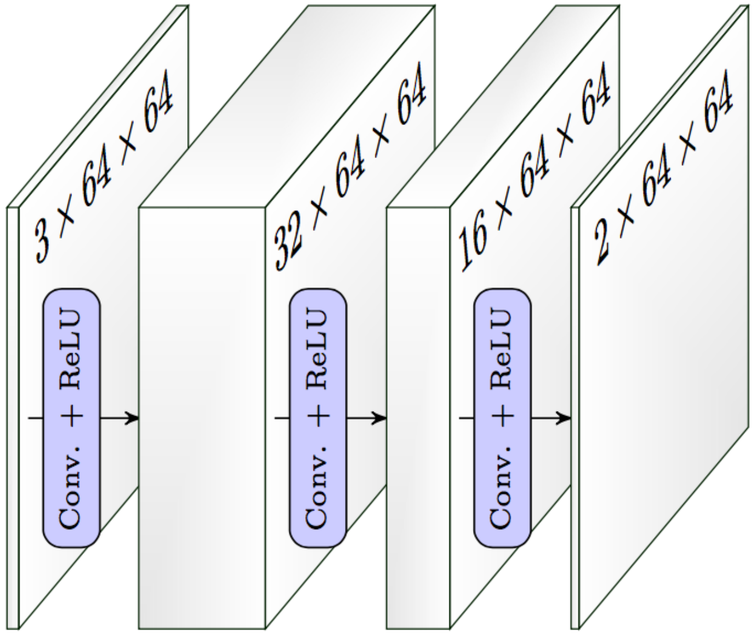


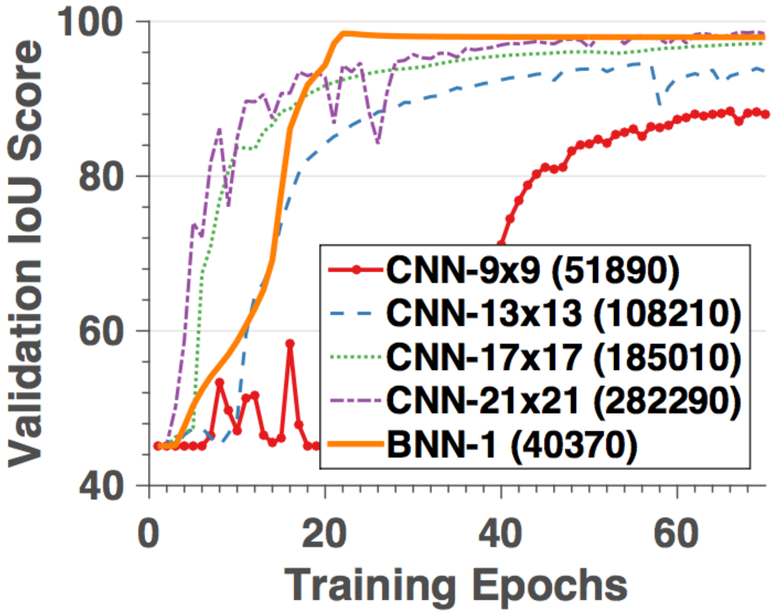
- High Dimensional and Sparse Convolution for CNNs

# Tile Segmentation 1/2

- Binary segmentation
- 64x64 images with 20x20 tile
- Random color and random position

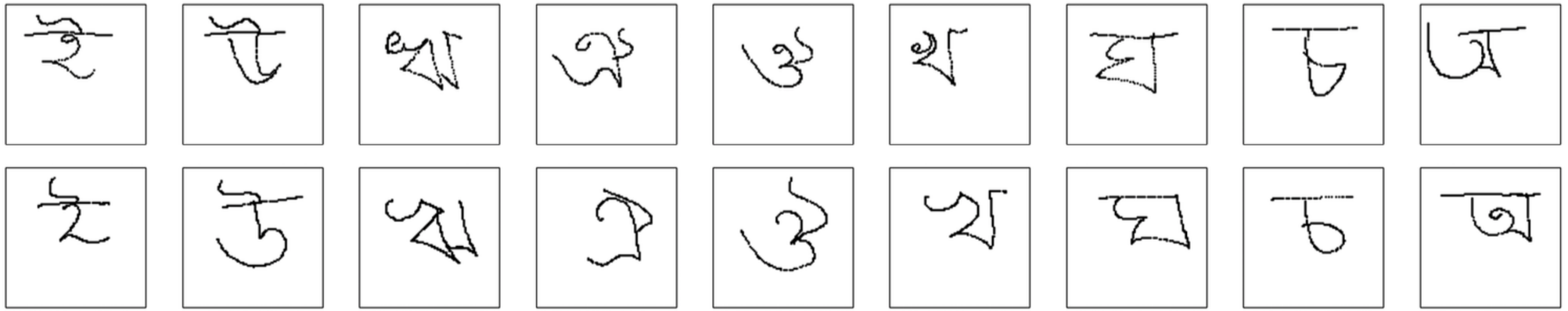
# Tile Segmentation 2/2

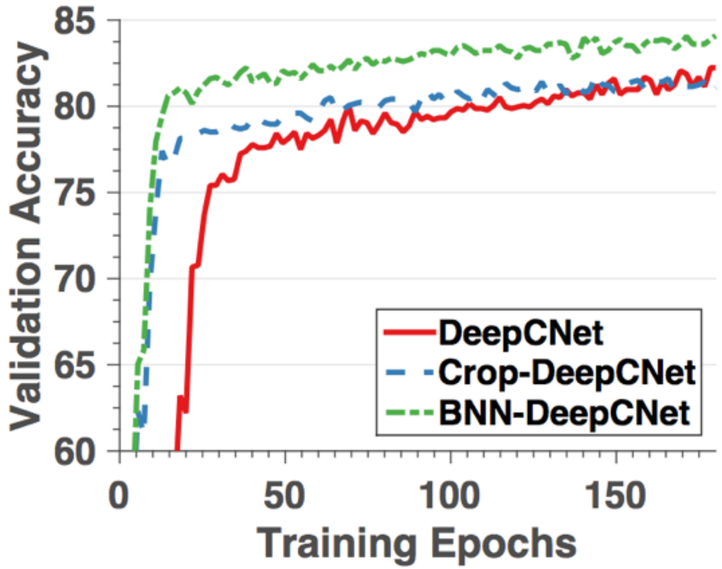



- All CNNs converge to almost perfect solution
- Spatial CNN versus Bilateral CNN
- BNN *see* the color, need to assign a label to the two groups

# Sparse Input Signal






| | LeNet | Crop-LeNet | BNN-LeNet | DeepC-Net | Crop-DeepCNet | BNN-DeepCNet |
|---|---|---|---|---|---|---|
| Validation | 59.29 | 68.67 | **75.05** | 82.24 | 81.88 | **84.15** |
| Test | 55.74 | 69.10 | **74.98** | 79.78 | 80.02 | **84.21** |

# 1) A simple Insight

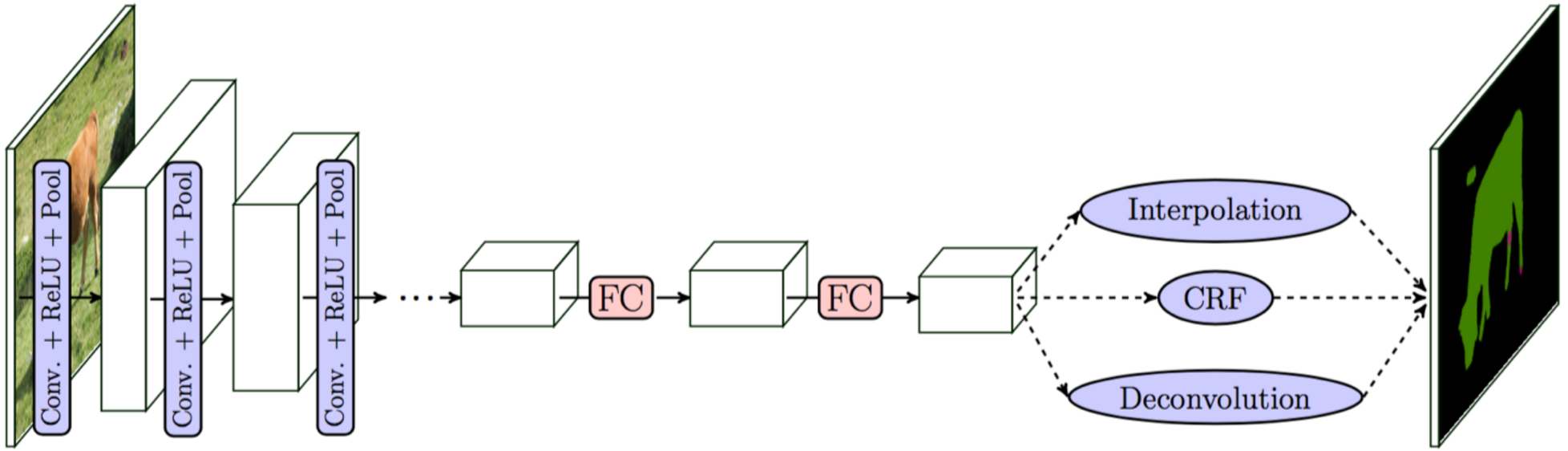
# 2) Consequences

Image Filtering

Bilateral CNNs

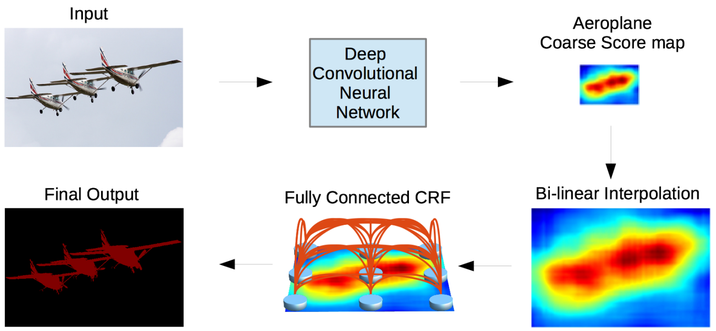
DenseCRFs

# Semantic Segmentation CNNs

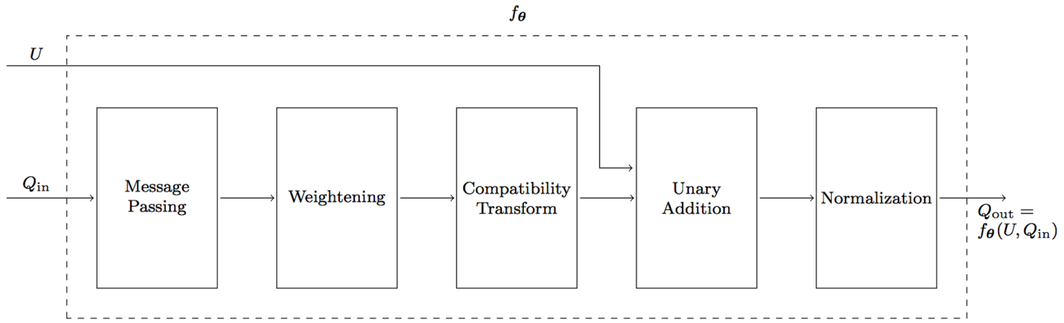



- Max-Pooling with striding reduces resolution!
- How to recover?

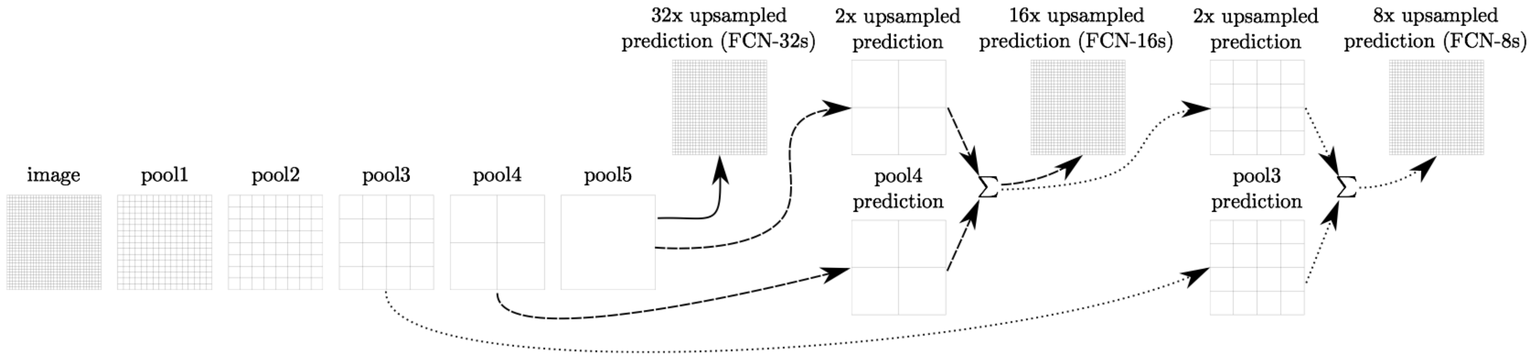
# Related Work (to undo max-pooling resolution change)



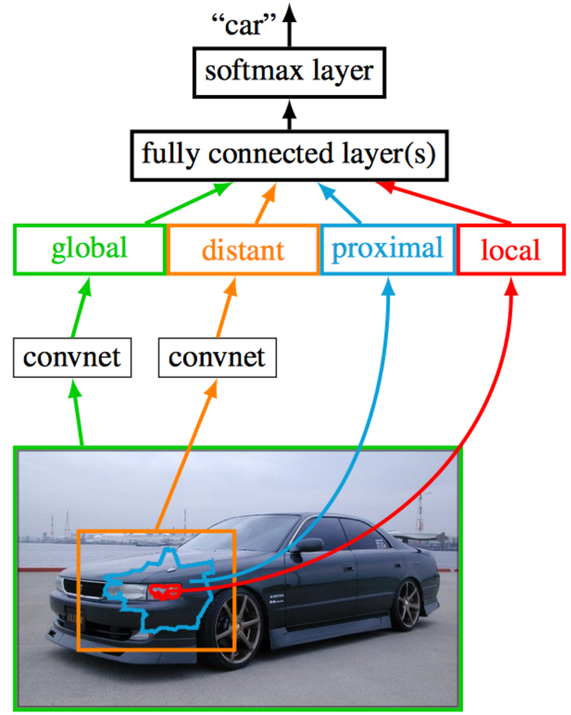

CRF + denseCRF
(Chen et al. ICLR 2015)




CRF + RNN
(Zheng et al. ICCV 15)




Deconvolutional Nets
(Long et al. 2014)




Zoom out nets
(Motasjabi et al. 15)

# “Dense CRF”

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# “Dense CRF”

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

- … with Gaussian Edge Potentials …

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) k(f_i, f_j) \qquad k(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)^\top \Sigma^{-1}(f_i - f_j)\right)$$

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# “Dense CRF”

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

- … with Gaussian Edge Potentials …

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) k(f_i, f_j) \qquad k(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)^\top \Sigma^{-1}(f_i - f_j)\right)$$

- … mean-field approximations can be computed using bilateral filtering!

$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{j \neq i} \sum_{x_j \in \mathcal{L}} \psi_p(x_i, x_j) q_j^t(x_j)\right)$$

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# “Dense CRF”

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

- … with Gaussian Edge Potentials …

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) k(f_i, f_j) \qquad k(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)^\top \Sigma^{-1}(f_i - f_j)\right)$$

- … mean-field approximations can be computed using bilateral filtering!

$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \boxed{\sum_{j \neq i} \sum_{x_j \in \mathcal{L}} \psi_p(x_i, x_j) q_j^t(x_j)}\right)$$

**Bilateral Filtering**

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# The Effect of Dense CRFs

Image


Ground Truth Segmentation

unaries only (here DeepLab)

with DenseCRF

# Generalized Class of Dense CRFs

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

- … with Gaussian Edge Potentials …

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) k(f_i, f_j) \qquad k(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)^\top \Sigma^{-1}(f_i - f_j)\right)$$

- … mean-field approximations can be computed using bilateral filtering!

$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{j \neq i} \sum_{x_j \in \mathcal{L}} \psi_p(x_i, x_j) q_j^t(x_j)\right)$$

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# Generalized Class of Dense CRFs

- For a densely connected CRF…

$$p(x|v) \propto \exp\left(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p(x_i, x_j)\right)$$

- … with Gaussian Edge Potentials …

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) k(f_i, f_j) \qquad k(f_i, f_j) = \exp\left(-\frac{1}{2}(f_i - f_j)^\top \Sigma^{-1}(f_i - f_j)\right)$$

- … mean-field approximations can be computed using bilateral filtering!

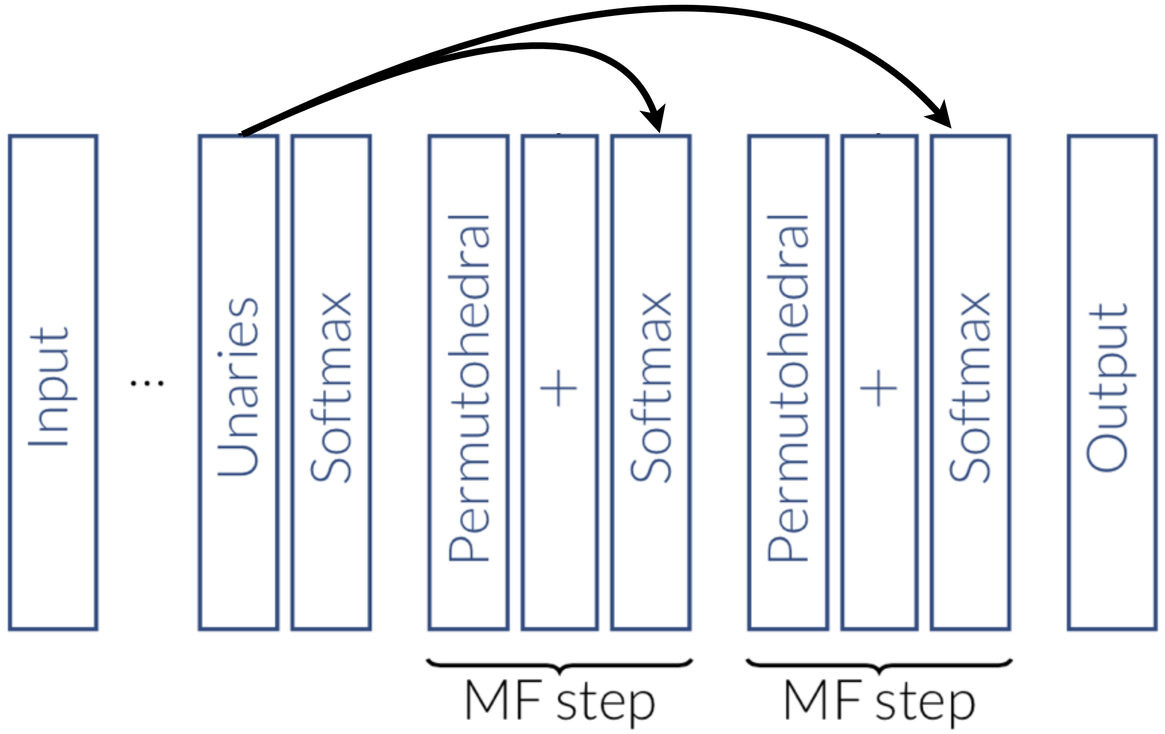
$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{j \neq i} \sum_{x_j \in \mathcal{L}} \psi_p(x_i, x_j) q_j^t(x_j)\right)$$

**almost arbitrary**

P. Krähenbühl and V. Koltun, *Efficient inference in fully connected CRFs with Gaussian edge potentials* , NIPS, 2011

# Learning Mean Field Kernels

$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{j \neq i} \sum_{x_j \in \mathcal{L}} \psi_p(x_i, x_j) q_j^t(x_j)\right)$$




J. Domke, *Learning graphical model parameters with approximate marginal inference*, PAMI, 2013

# Segmentation Empirical Results

| | + MF-1step | + MF-2 step | + *loose* MF-2 step |
|---|---|---|---|
| Semantic segmentation (IoU) - CNN [11]: 72.08 / 66.95 | | | |
| Gauss CRF | +2.48 | +3.38 | +3.38 / +3.00 |
| Learned CRF | +2.93 | +3.71 | **+3.85 / +3.37** |
| Material segmentation (Pixel Accuracy) - CNN [7]: 67.21 / 69.23 | | | |
| Gauss CRF | +7.91 / +6.28 | +9.68 / +7.35 | +9.68 / +7.35 |
| Learned CRF | +9.48 / +6.23 | +11.89 / +6.93 | **+11.91 / +6.93** |

- DeepLab as base model
- Pascal VOC12 validation/test: IoU
- Material In Context (MINC) pixel /class accuracy

Chen et al., *Semantic image segmentation with deep convolutional nets and fully connected CRFs*, ICLR 2015
S. Bell, P. Upchurch, N. Snavely, and K. Bala , *Material recognition in the wild with the materials in context database*, CVRP 2015

# 1) A simple Insight

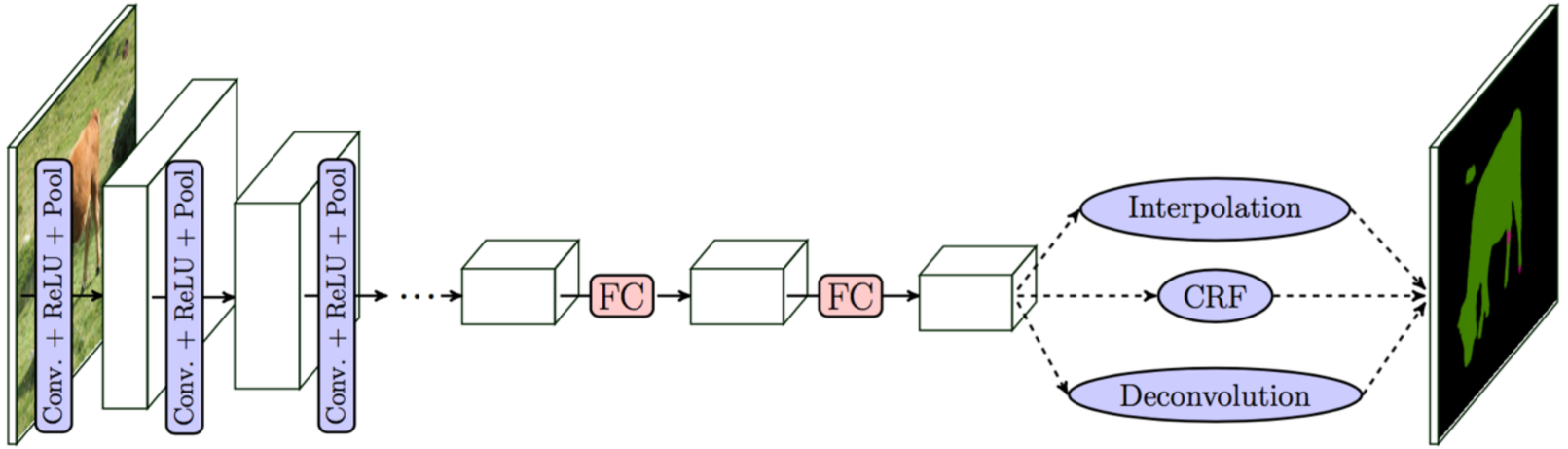
# 2) Consequences

Image Filtering

Bilateral CNNs

DenseCRFs

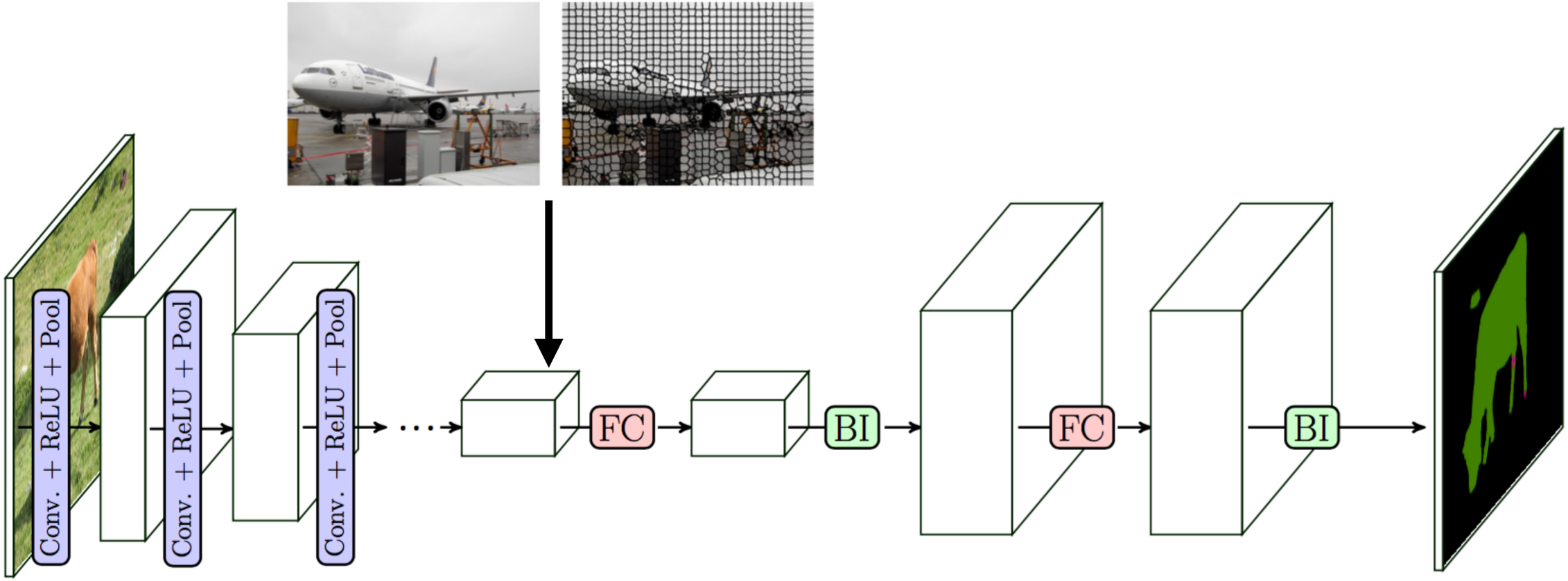
# Semantic Segmentation




- Dense CRF "on top" improves because it encodes prior information not yet learned by the network
- How to **encode inside** the network?
- How to recover full resolution?

# Idea 1 : Use superpixels

$$(f_i, z_i), f_i \in \mathbb{R}^5, z_i \in \mathbb{R}^{4096}$$



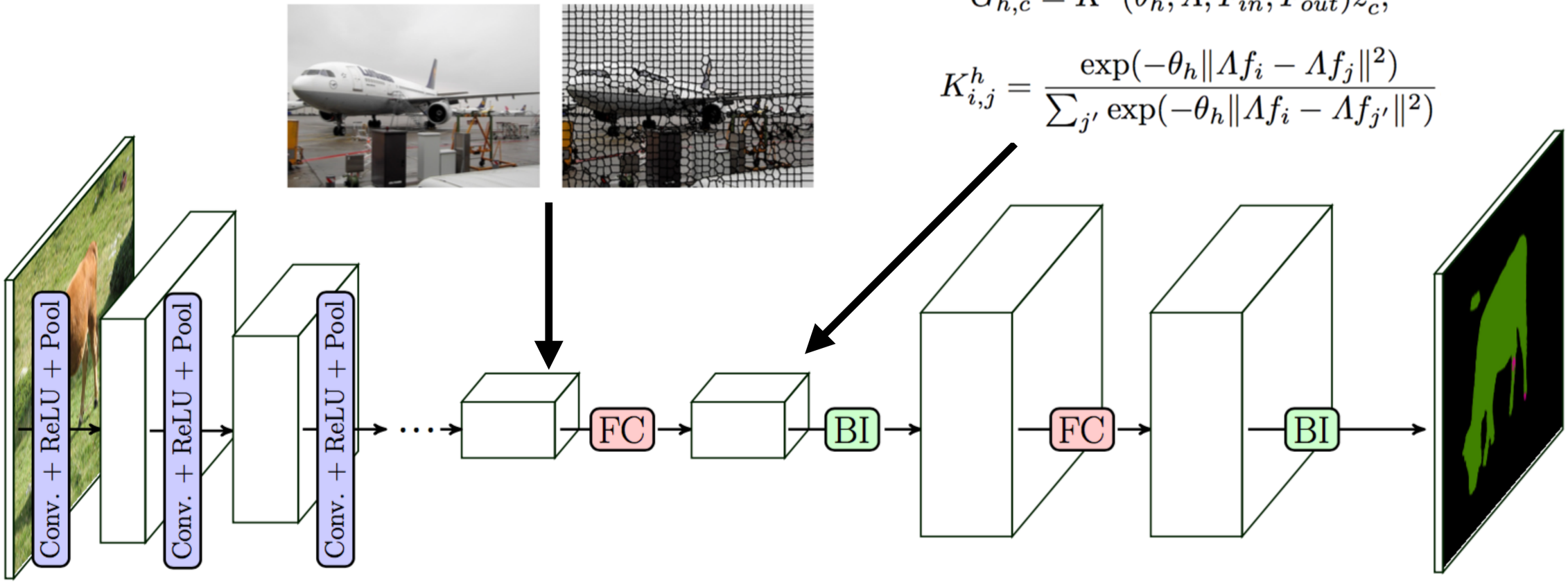
# Idea 2 : Filter network activations

$(f_i, z_i), f_i \in \mathbb{R}^5, z_i \in \mathbb{R}^{4096}$

Gauss Filter

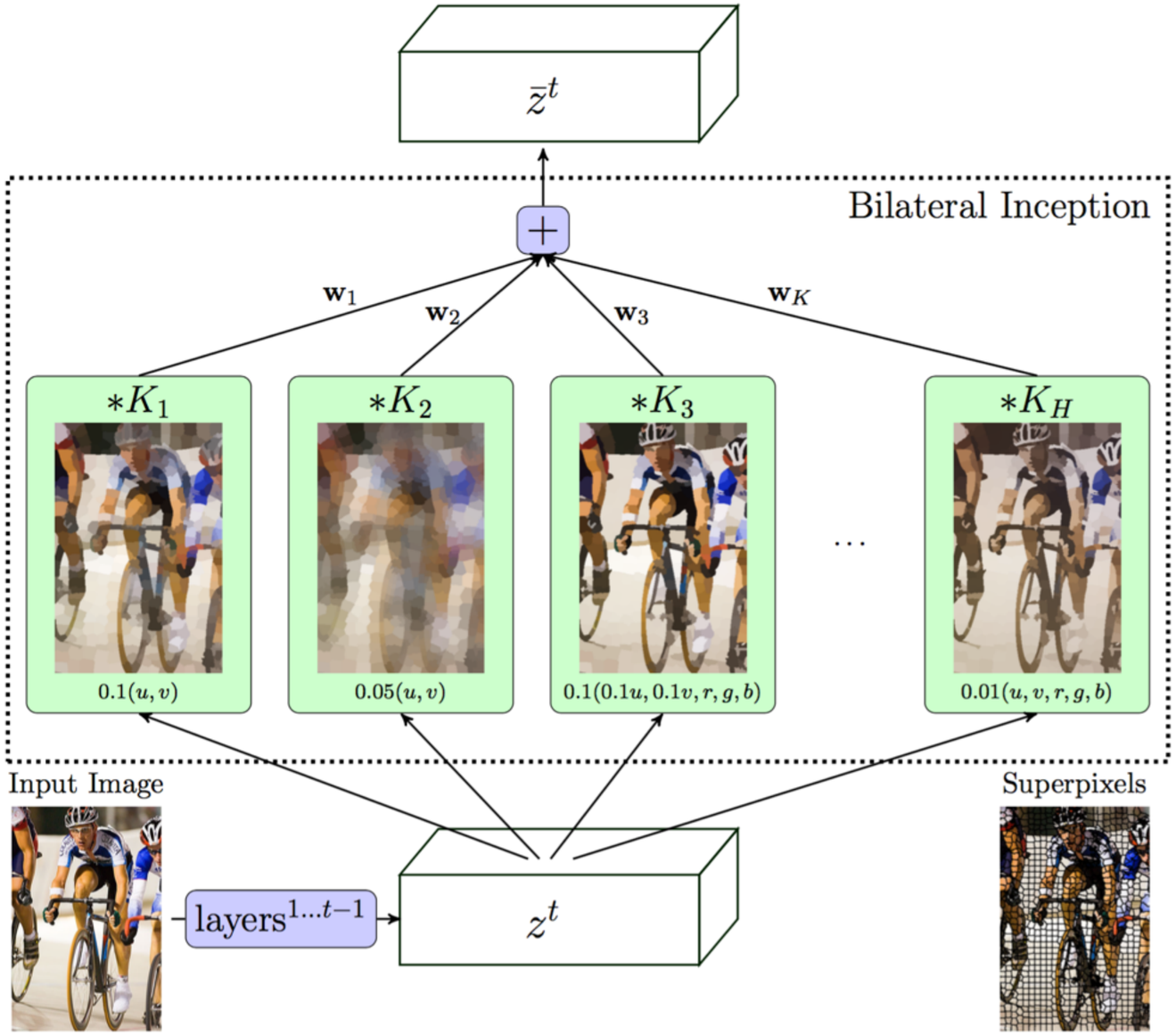
$$G_{h,c} = K^h(\theta_h, \Lambda, F_{in}, F_{out}) z_c^t,$$

$$K^h_{i,j} = \frac{\exp(-\theta_h \|\Lambda f_i - \Lambda f_j\|^2)}{\sum_{j'} \exp(-\theta_h \|\Lambda f_i - \Lambda f_{j'}\|^2)}$$




Contrary to DenseCRF that filters the output

# Bilateral Inception Module



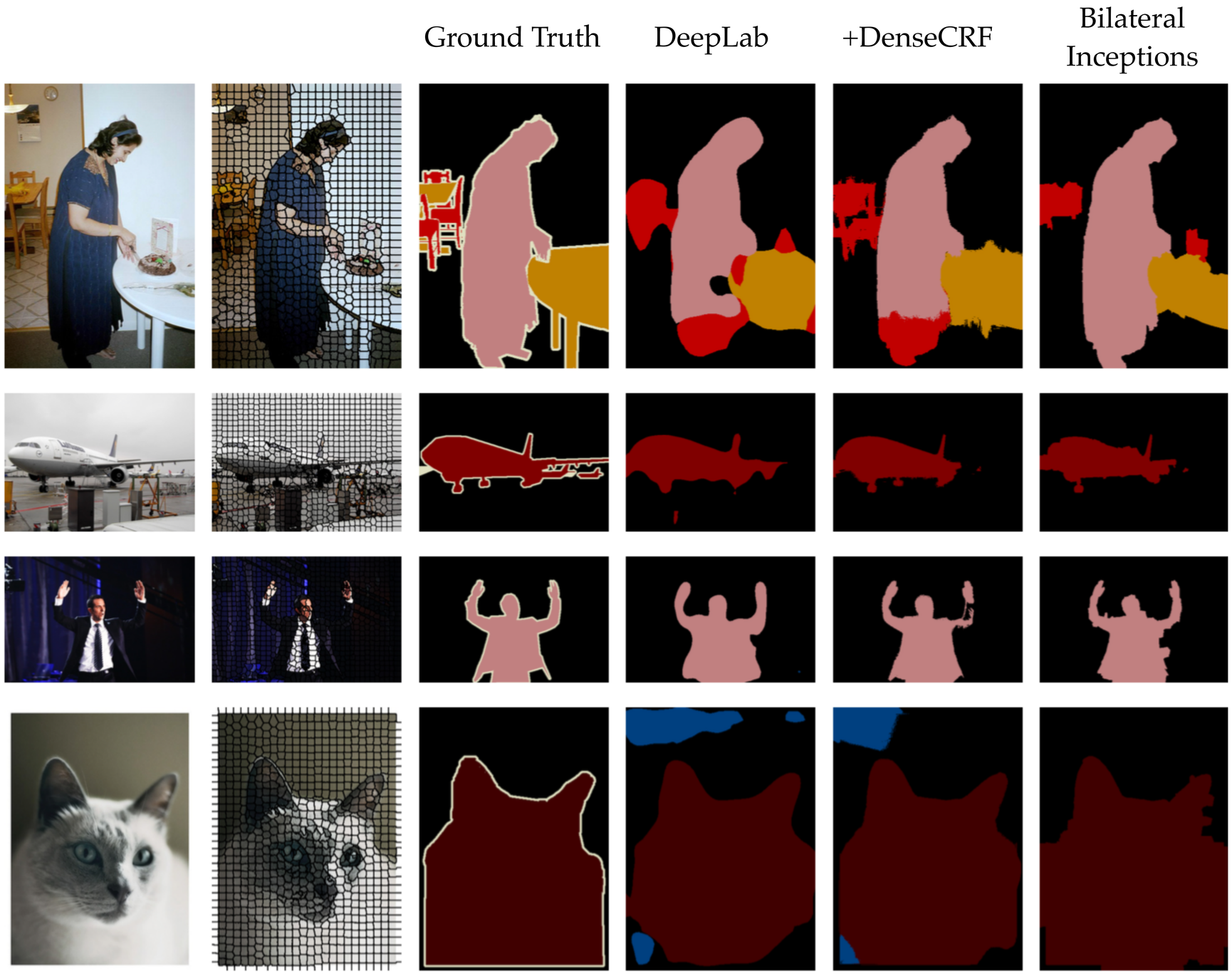

Gaussian Filter …

$$G_{h,c} = K^h(\theta_h, \Lambda, F_{in}, F_{out}) z_c^t,$$

$$K_{i,j}^h = \frac{\exp(-\theta_h \|\Lambda f_i - \Lambda f_j\|^2)}{\sum_{j'} \exp(-\theta_h \|\Lambda f_i - \Lambda f_{j'}\|^2)}$$

… on multiple scales

$$\bar{z}_c^t = \sum_h w_{h,c} * G_{h,c}$$


Ground Truth
DeepLab
+DenseCRF
Bilateral Inceptions

# Empirical Results (see paper for more)

| **Model** | *IoU* | *Runtime* |
|---|---|---|
| DeepLab [3] | 68.9 | 135 |
| With inception modules<br>$BI_6(2)$ - module training | 70.8 | +20 |

# Empirical Results (see paper for more)

| **Model** | *IoU* | *Runtime* |
|---|---|---|
| DeepLab [3] | 68.9 | 135 |
| With inception modules | | |
| $BI_6(2)$ - module training | 70.8 | +20 |
| $BI_6(2)$ - joint training | 71.5 | +20 |

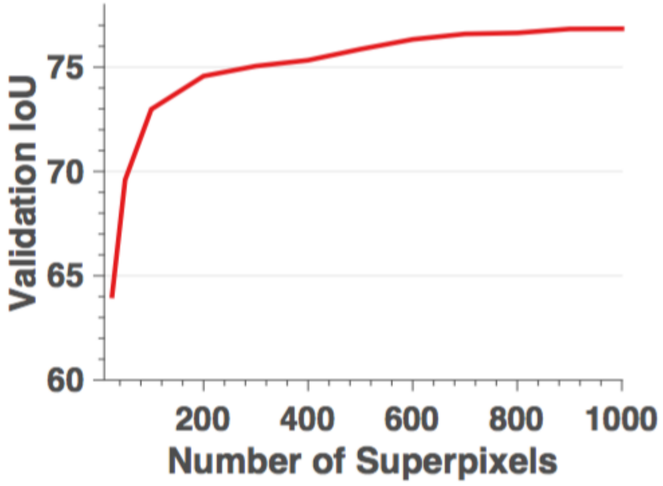
# Empirical Results (see paper for more)

| **Model** | *IoU* | *Runtime* |
|---|---|---|
| DeepLab [3] | 68.9 | 135 |
| With inception modules | | |
| $BI_6(2)$ - module training | 70.8 | +20 |
| $BI_6(2)$ - joint training | 71.5 | +20 |
| $BI_6(6)$ | 72.9 | +45 |
| $BI_7(6)$ | 73.1 | +50 |
| $BI_8(10)$ | 72.0 | +30 |

# Empirical Results (see paper for more)

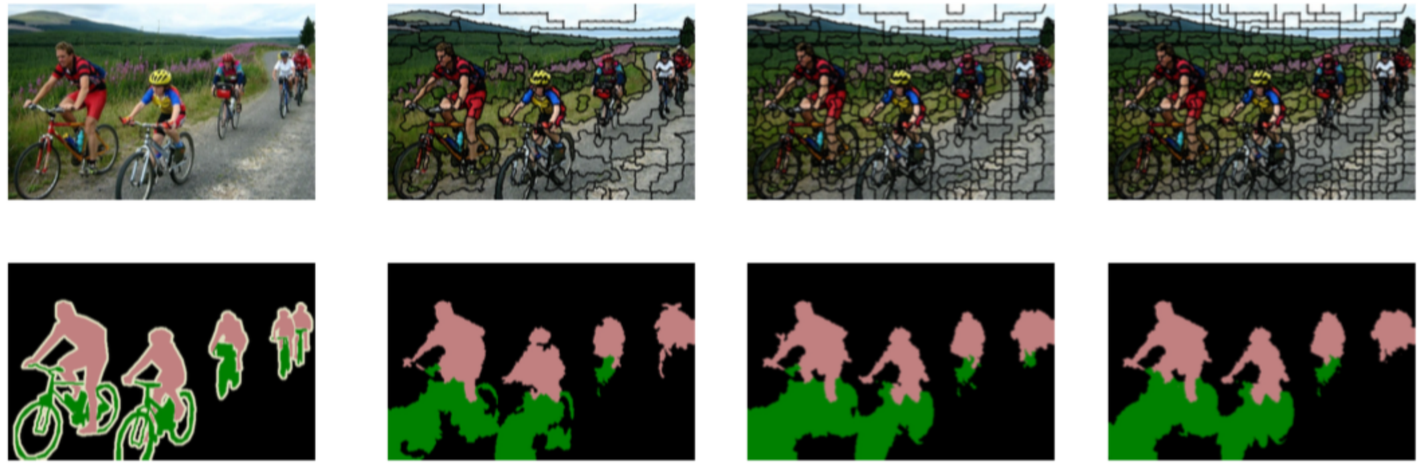
| **Model** | *IoU* | *Runtime* |
|---|---|---|
| DeepLab [3] | 68.9 | 135 |
| With inception modules | | |
| $BI_6(2)$ - module training | 70.8 | +20 |
| $BI_6(2)$ - joint training | 71.5 | +20 |
| $BI_6(6)$ | 72.9 | +45 |
| $BI_7(6)$ | 73.1 | +50 |
| $BI_8(10)$ | 72.0 | +30 |
| $BI_6(2)$-$BI_7(6)$ | **73.6** | +35 |
| $BI_7(6)$-$BI_8(10)$ | 73.4 | +55 |
| DeepLab-CRF [3] | 72.7 | +830 |
| DeepLab-MSc-CRF [3] | **73.6** | +880 |
| DeepLab-EdgeNet [6] | 71.7 | +30 |
| DeepLab-EdgeNet-CRF [6] | **73.6** | +860 |

# Adaptive FC Layout




(trained with)

200 600 1000

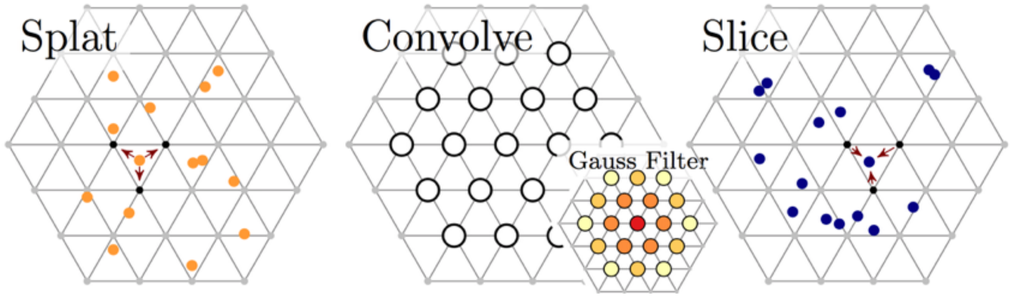


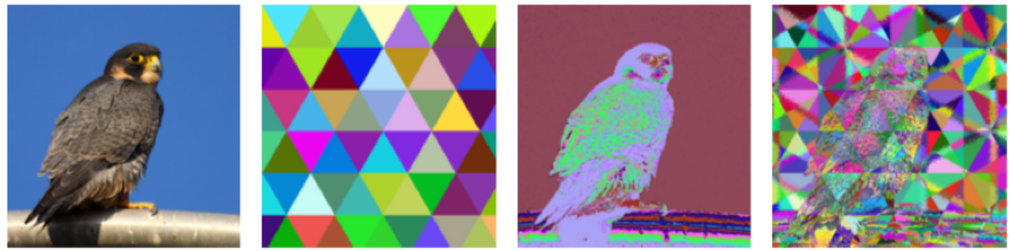
- Vary the number of superpixels after training

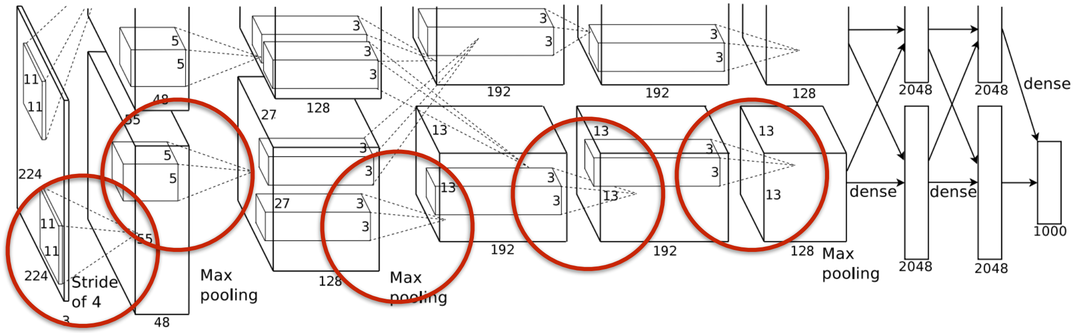
# Conclusion

- Presented: Generalized Bilateral Filtering
- Is an: Edge-aware CNN and General Dense CRFs
  - But: You do not need a (Dense)CRF!

- Why always grid layouts for CNNs?
- Todo: use different lattices inside a CNN
- Todo: learn the embedding



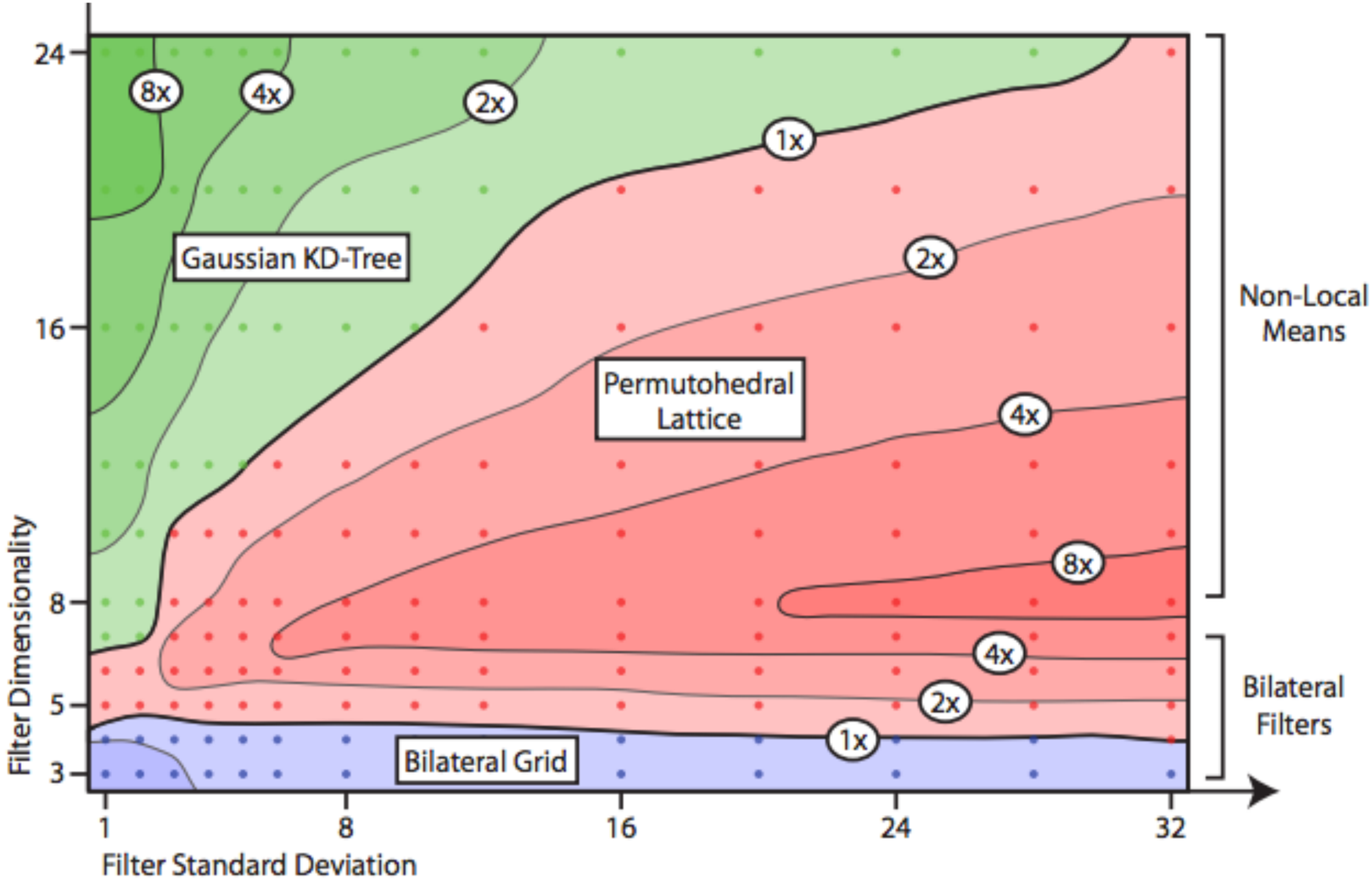


# Runtime (caffe implementation)

| Dim.-Features | d-dim caffe | BCL |
|---|---|---|
| 2D-$(x, y)$ | **3.3 ± 0.3 / 0.5± 0.1** | 4.8 ± 0.5 / 2.8 ± 0.4 |
| 3D-$(r, g, b)$ | 364.5 ± 43.2 / 12.1 ± 0.4 | **5.1 ± 0.7 / 3.2 ± 0.4** |
| 4D-$(x, r, g, b)$ | 30741.8 ± 9170.9 / 1446.2 ± 304.7 | **6.2 ± 0.7 / 3.8 ± 0.5** |
| 5D-$(x, y, r, g, b)$ | out of memory | **7.6 ± 0.4 / 4.5 ± 0.4** |

- CPU/GPU runtime (ms)
- 50 filters averaged over 1000 images
- BCL includes splat/splice (could be re-used)
- no full GPU implementation yet

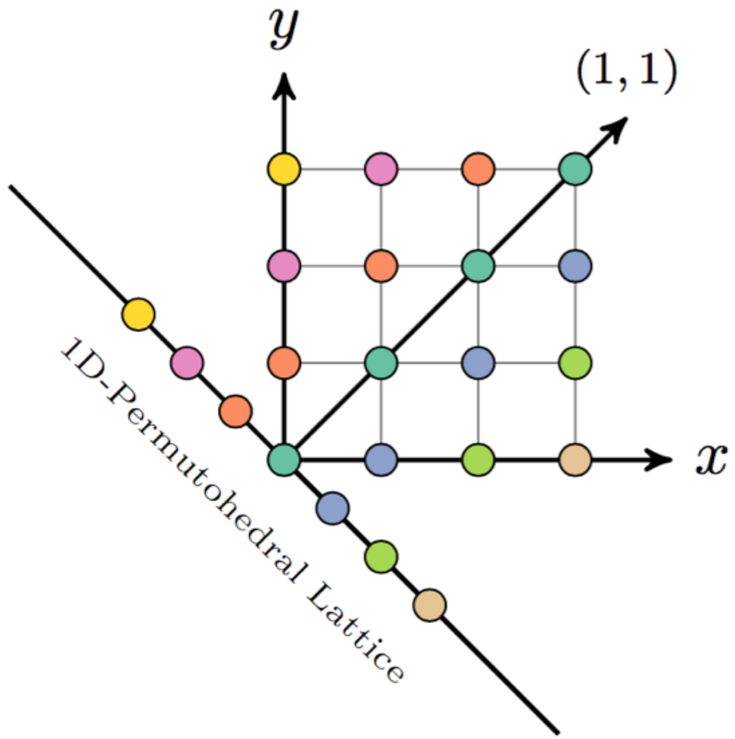
# Permutohedral Lattice Convolutions




A. Adams, J. Baek, M. Davis, *Fast High-Dimensional Filtering Using the Permutohedral Lattice* , Eurographics, 2009

# The Permutohedral Lattice

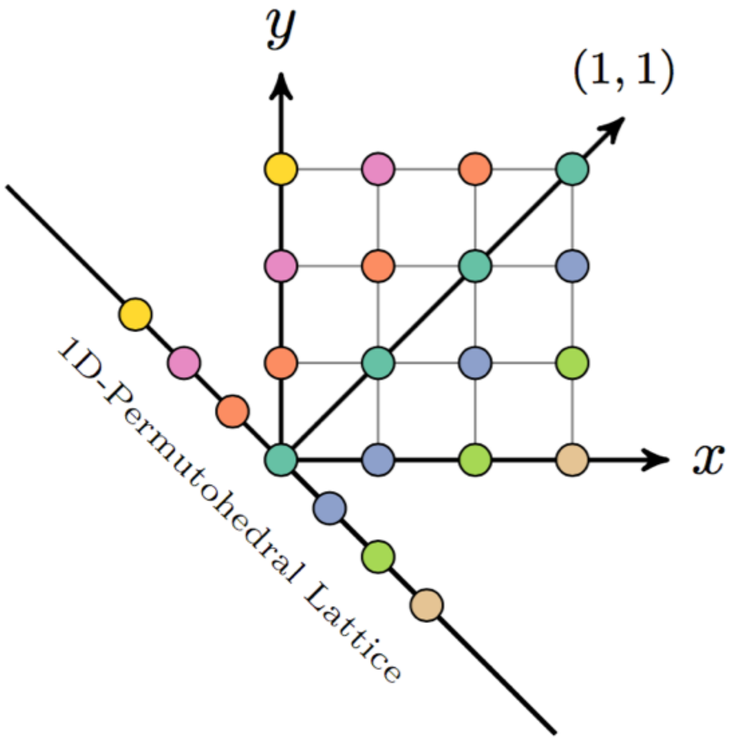
d=2



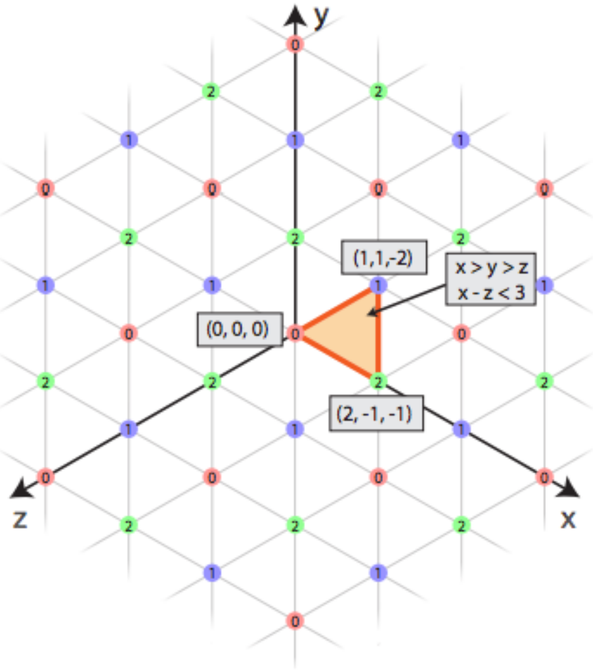

A. Adams, J. Baek, M. Davis, *Fast High-Dimensional Filtering Using the Permutohedral Lattice* , Eurographics, 2009

# The Permutohedral Lattice

d=2




d=3




A. Adams, J. Baek, M. Davis, *Fast High-Dimensional Filtering Using the Permutohedral Lattice* , Eurographics, 2009

# Segmentation Results

## Material Classification

| **Model** | Class / Total accuracy | Runtime |
|---|---|---|
| Alexnet CNN | 55.3 / 58.9 | 300 |
| $BI_7(2)$-$BI_8(6)$ | 67.7 / 71.3 | 410 |
| $BI_7(6)$-$BI_8(6)$ | **69.4 / 72.8** | 470 |
| AlexNet-CRF | 65.5 / 71.0 | 3400 |

## Cityscapes

| **Model** | IoU (Half-res.) | IoU (Full-res.) | Runtime |
|---|---|---|---|
| DeepLab CNN | 62.2 | 65.7 | 0.3 |
| $BI_6(2)$ | 62.7 | 66.5 | 5.7 |
| $BI_6(2)$-$BI_7(6)$ | **63.1** | **66.9** | 6.1 |
| DeepLab-CRF | 63.0 | 66.6 | 6.9 |