Integrating Mathematical Morphology within Deep Convolutional Neural Networks

Alexandre Kirszenberg

September 30th, 2019

Given an image, detect the outline of objects automatically.

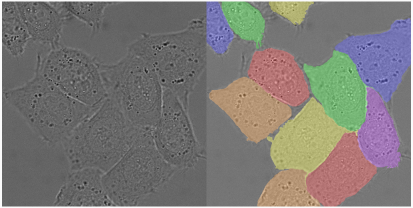
Chen et al., “Rethinking Atrous Convolution for Semantic Image Segmentation”

An introduction to Image Segmentation

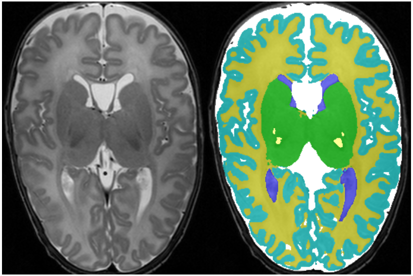
Given an image, detect the outline of objects automatically.

Chen et al., "Rethinking Atrous Convolution for Semantic Image Segmentation"

Image Segmentation: Application to Medical Imaging



HeLa cells



Axial adult brain

Ronneberger, Fischer, and Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”

Xu, Géraud, and Bloch, “From neonatal to adult brain MR image segmentation in a few seconds using 3D-like fully convolutional network and transfer learning”

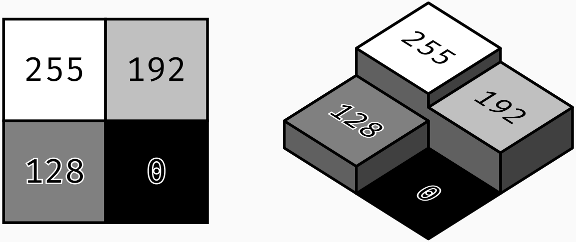
- The theory of Mathematical Morphology considers images as landscapes.[1]

[1] Géraud, *A Quick Tour of Mathematical Morphology*.

- The theory of Mathematical Morphology considers images as landscapes.[1]
- The value of the pixel at position (x, y) represents its elevation.

[1] Géraud, *A Quick Tour of Mathematical Morphology*.

- The theory of Mathematical Morphology considers images as landscapes.[1]
- The value of the pixel at position (x, y) represents its elevation.



[1] Géraud, *A Quick Tour of Mathematical Morphology*.

Morphological Filters

- Mathematical Morphology defines a variety of morphological filters (or operators).

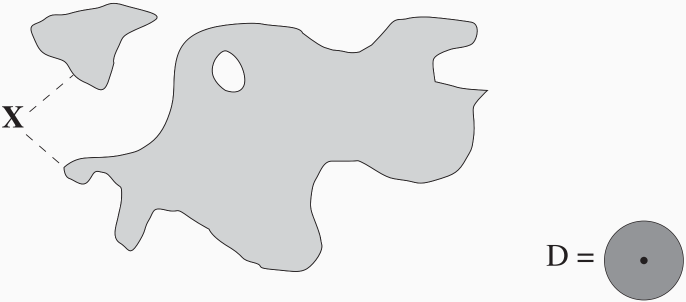
Morphological Filters

- Mathematical Morphology defines a variety of morphological filters (or operators).
- These filters are the composition of two operations:
    - sum $+$
    - infimum $\wedge$ (or supremum $\vee$)

Morphological Filters

- Mathematical Morphology defines a variety of morphological filters (or operators).
- These filters are the composition of two operations:
    - sum $+$
    - infimum $\wedge$ (or supremum $\vee$)
- They take a structuring element as parameter.

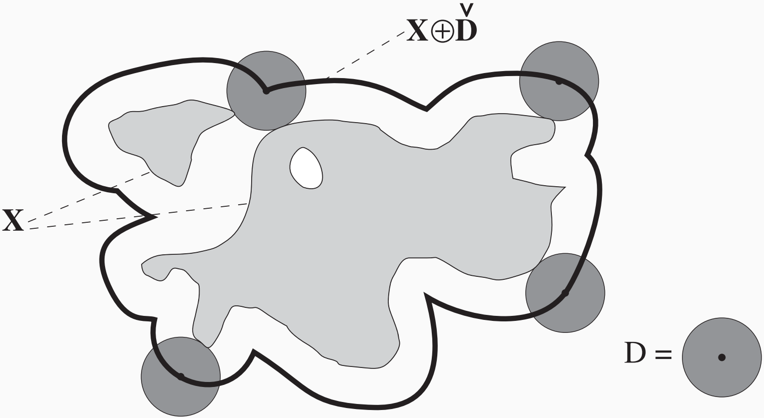
Morphological Filters: an example



A set X and a structuring element D

Serra and Vincent, "An overview of morphological filtering"

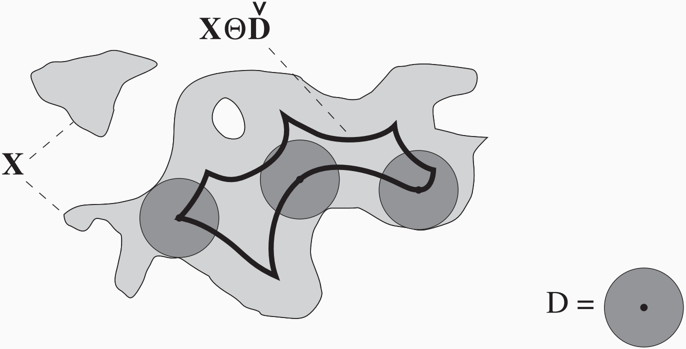
Morphological Filters: an example



The dilation of *X* by *D*

Serra and Vincent, "An overview of morphological filtering"

Morphological Filters: an example

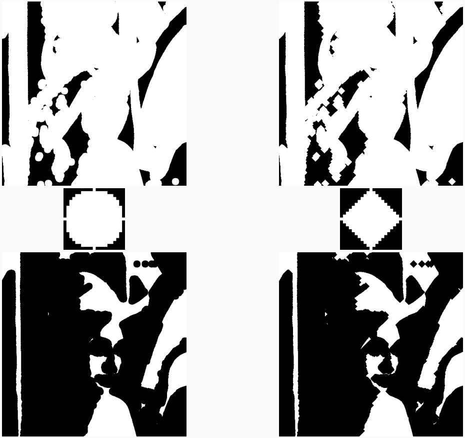


The erosion of *X* by *D*

Serra and Vincent, "An overview of morphological filtering"

Morphological Filters: with an actual image

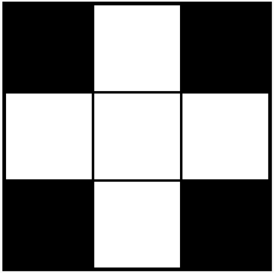
Dilation δ_C Dilation δ_D



Erosion ϵ_C Erosion ϵ_D

From Binary Dilation to Grayscale Dilation

$$(f \oplus b)(x) = \sup_{y \in E} [f(y) + b(x - y)]$$

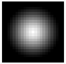


0	1	0
1	1	1
0	1	0

-∞	0	-∞
0	0	0
-∞	0	-∞

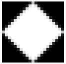
Grayscale Filters

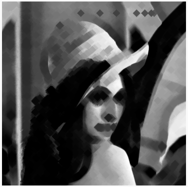
Dilation δ_C



Erosion ϵ_C

Dilation δ_D





Erosion ϵ_D

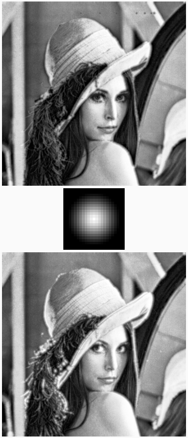
Higher Order Filters

We can combine these filters to create higher order operators:

- The opening $\gamma = \delta \circ \epsilon$.
- The closing $\phi = \epsilon \circ \delta$.

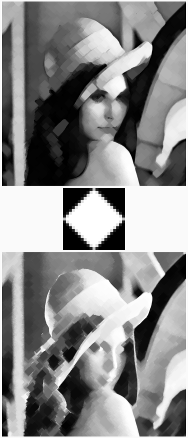
Higher Order Filters: an example

Opening γ_C



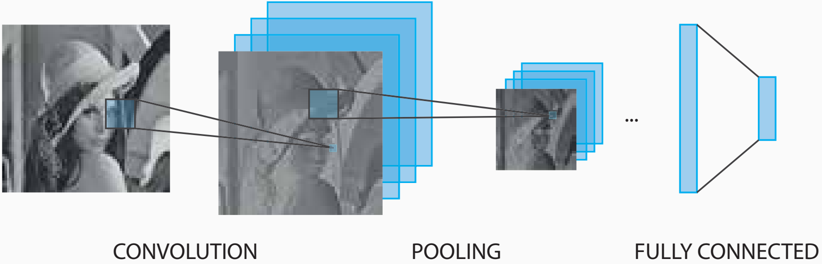
Closing ϕ_C

Opening γ_D



Closing ϕ_D

Convolutional Neural Networks



$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau) \, d\tau$$

Lecun et al., "Gradient-based learning applied to document recognition"

Masci, Angulo, and Schmidhuber, "A Learning Framework for Morphological Operators using Counter-Harmonic Mean"

A Fixed Dilation Layer

- Rather straightforward to implement with Tensorflow.[2]

[2] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[3] Dougherty, *An introduction to morphological image processing*.

A Fixed Dilation Layer

- Rather straightforward to implement with Tensorflow.[2]
- Tensorflow implements Grayscale Dilation[3]:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

[2] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[3] Dougherty, *An introduction to morphological image processing*.

A Fixed Dilation Layer

- Rather straightforward to implement with Tensorflow.[2]
- Tensorflow implements Grayscale Dilation[3]:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

- We first experimented with fixed binary structuring elements (e.g. a vertical cross, an X).

[2] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[3] Dougherty, *An introduction to morphological image processing*.

A Fixed Dilation Layer

- Rather straightforward to implement with Tensorflow.[2]
- Tensorflow implements Grayscale Dilation[3]:

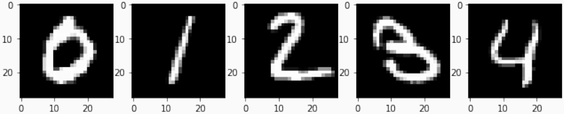
$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

- We first experimented with fixed binary structuring elements (e.g. a vertical cross, an X).
- As such, we had to apply the previously mentioned transformation to our structuring elements.

[2] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[3] Dougherty, *An introduction to morphological image processing*.

The Data

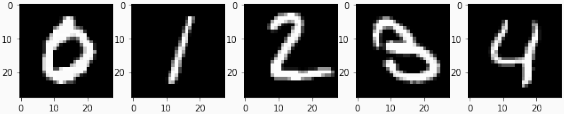
- For the purpose of these experiments, we used the MNIST Handwritten Digits[4] dataset.



[4]LeCun and Cortes, "MNIST handwritten digit database".

The Data

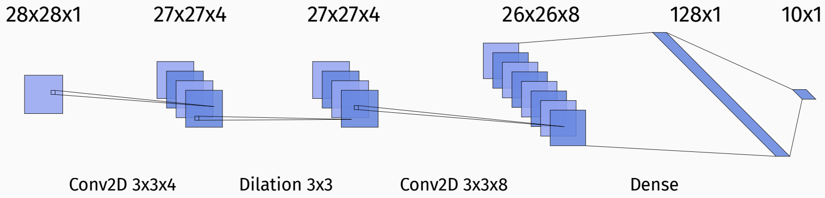
- For the purpose of these experiments, we used the MNIST Handwritten Digits[4] dataset.



- 28×28 grayscale, $60\,000$ training images, $10\,000$ test images.

[4] LeCun and Cortes, "MNIST handwritten digit database".

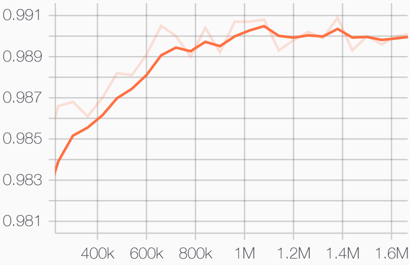
Testing a Dilation Layer on MNIST: Architecture



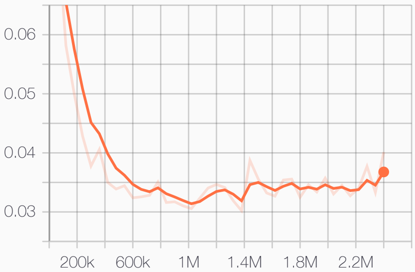
Also tested with a single level of convolution + dilation.

Testing a Dilation Layer on MNIST: Training

Trained over 40 epochs, with a batch size of 128.



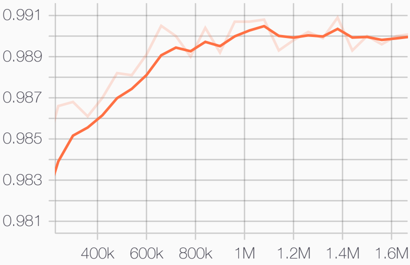
Accuracy on the test set $a_{max} = 0.9914$



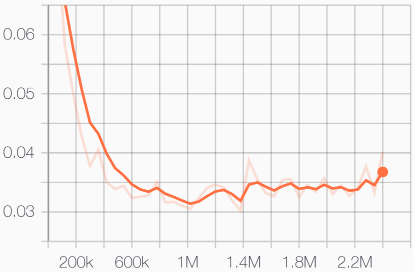
Loss on the test set $l_{min} = 0.0325$

Testing a Dilation Layer on MNIST: Training

Trained over 40 epochs, with a batch size of 128.



Accuracy on the test set $a_{max} = 0.9914$



Loss on the test set $l_{min} = 0.0325$

A more classical architecture with a MaxPooling layer reaches similar scores: $a_{max} = 0.9923$, $l_{min} = 0.0252$ (~150 epochs).

Learning a Structuring Element

- Tensorflow[5] itself already implements the operations of Grayscale Dilation and Erosion[6]:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

[5] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[6] Dougherty, *An introduction to morphological image processing*.

Learning a Structuring Element

- Tensorflow[5] itself already implements the operations of Grayscale Dilation and Erosion[6]:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

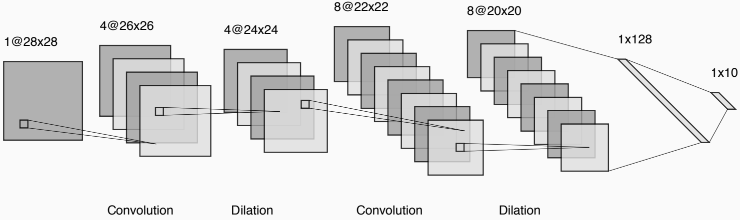
- Creating a grayscale dilation layer is mainly just a call to `Layer::add_weight` and `tf.nn.dilation2d`.

[5] Abadi et al., “TensorFlow: A system for large-scale machine learning”.
[6] Dougherty, *An introduction to morphological image processing*.

- Tensorflow[5] itself already implements the operations of Grayscale Dilation and Erosion[6]:

$$(f \oplus b)(x) = \sup_{y \in E}[f(y) + b(x - y)]$$

- Creating a grayscale dilation layer is mainly just a call to `Layer::add_weight` and `tf.nn.dilation2d`.



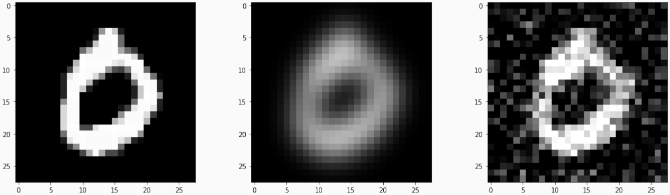
[5] Abadi et al., "TensorFlow: A system for large-scale machine learning".
[6] Dougherty, *An introduction to morphological image processing*.

Adding Destructive Filters to the Mix

- We experimented with adding destructive filters to the input.

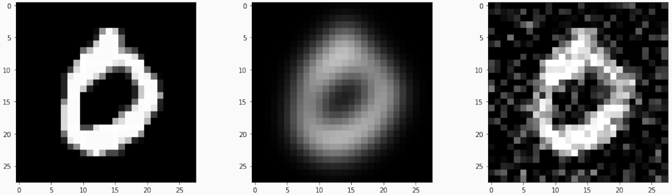
Adding Destructive Filters to the Mix

- We experimented with adding destructive filters to the input.
- For instance, with gaussian blur and/or gaussian noise:



Adding Destructive Filters to the Mix

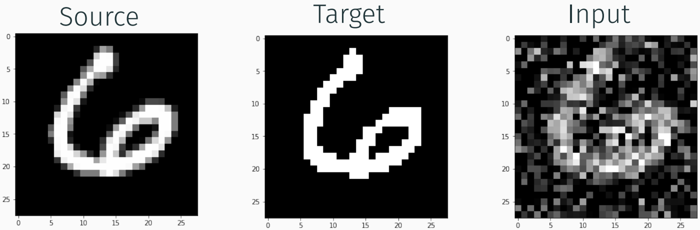
- We experimented with adding destructive filters to the input.
- For instance, with gaussian blur and/or gaussian noise:



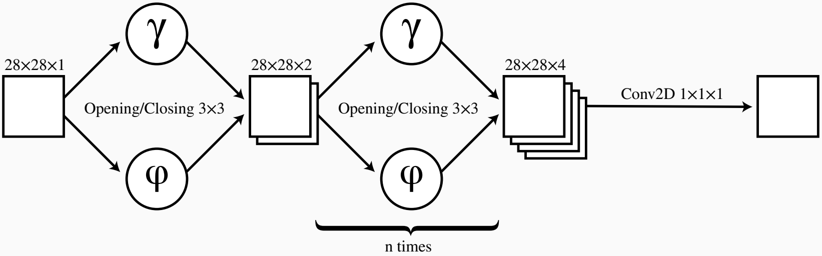
- Dilation layers do not appear to outperform similarly-shaped convolution layers.

  Val. acc. of 0.9792 vs 0.9819 after ~20 epochs, resp.

We keep our MNIST dataset, but this time we want to classify each input pixel into binary classes.



We can already get some results with very few parameters.



Total number of parameters: 135

Comparing Results

- This architecture reached a F1-score[7] of 0.8304 (precision 0.8469, recall 0.8145).

[7] Rijsbergen, *Information retrieval*.

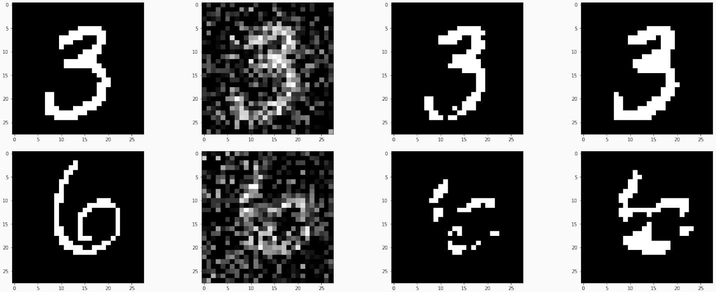
Comparing Results

- This architecture reached a F1-score[7] of 0.8304 (precision 0.8469, recall 0.8145).
- However, a similar fully convolutional model using approximately the same number of parameters (147 vs 135) reached a F1-Score of 0.8542 (precision 0.8400, recall 0.8690).

[7] Rijsbergen, *Information retrieval.*

Comparing Results

- This architecture reached a F1-score[7] of 0.8304 (precision 0.8469, recall 0.8145).
- However, a similar fully convolutional model using approximately the same number of parameters (147 vs 135) reached a F1-Score of 0.8542 (precision 0.8400, recall 0.8690).



[7] Rijsbergen, *Information retrieval.*

Conclusions

- These experiments showed no significant benefit in leveraging morphological filters within the structure of standard convolutional neural networks.

Conclusions

- These experiments showed no significant benefit in leveraging morphological filters within the structure of standard convolutional neural networks.
- However, the “toy” challenge we set ourselves does not map to a real world problem.

Conclusions

- These experiments showed no significant benefit in leveraging morphological filters within the structure of standard convolutional neural networks.
- However, the "toy" challenge we set ourselves does not map to a real world problem.
- There is still much experimentation to be done in the field!

PConv

- We started experimenting with PConv layers[8]:

$$PConv(f; w, P)(x) = \frac{(f^{P+1} * w)(x)}{(f^P * w)(x)}$$

[8] Masci, Angulo, and Schmidhuber, "A Learning Framework for Morphological Operators using Counter-Harmonic Mean".

PConv

- We started experimenting with PConv layers[8]:

$$PConv(f; w, P)(x) = \frac{(f^{P+1} * w)(x)}{(f^P * w)(x)}$$

- These layers learn not only the filter, but also the morphological operation:

[8] Masci, Angulo, and Schmidhuber, “A Learning Framework for Morphological Operators using Counter-Harmonic Mean”.

PConv

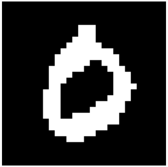
- We started experimenting with PConv layers[8]:

$$PConv(f; w, P)(x) = \frac{(f^{P+1} * w)(x)}{(f^P * w)(x)}$$

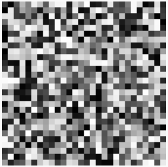
- These layers learn not only the filter, but also the morphological operation:
  - $P < 0$ is a pseudo-erosion.
  - $P > 0$ is a pseudo-dilation.

[8] Masci, Angulo, and Schmidhuber, “A Learning Framework for Morphological Operators using Counter-Harmonic Mean”.

A first PConv result

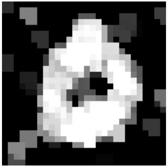


Target



Input

$P = -0.6663$



$P = 4.2216$

$P = -1.2166$

Training PConv layers

- PConv layers are harder to train than our fixed-operation layers...

Training PConv layers

- PConv layers are harder to train than our fixed-operation layers…
- …with a bunch of edge cases involving `NaN`.

Training PConv layers

- PConv layers are harder to train than our fixed-operation layers…
- …with a bunch of edge cases involving `NaN`.
- The original paper proposes alternating between learning P and the weights w.

What's next?

- Move to a real problem, e.g. dHCP.[9]

[9] http://www.developingconnectome.org/project/

What's next?

- Move to a real problem, e.g. dHCP.[9]
- This will allow us to experiment with integrating morphological filters within much more complex architectures.

[9] http://www.developingconnectome.org/project/

References

Abadi, Martin et al. “TensorFlow: A system for large-scale machine learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: `https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf`.

Chen, Liang-Chieh et al. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1706.05587 (2017). arXiv: `1706.05587`. URL: `http://arxiv.org/abs/1706.05587`.

Dougherty, E.R. *An introduction to morphological image processing*. Tutorial texts in optical engineering. SPIE Optical Engineering Press, 1992. URL: `https://books.google.fr/books?id=1kvxAAAAMAAJ`.

Géraud, Thierry. *A Quick Tour of Mathematical Morphology*. Huazhong University of Science & Technology and Wuhan University. Sept. 2017.

LeCun, Yann and Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: `http://yann.lecun.com/exdb/mnist/`.

Lecun, Y. et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 0018-9219. DOI: `10.1109/5.726791`.

Masci, Jonathan, Jesús Angulo, and Jürgen Schmidhuber. "A Learning Framework for Morphological Operators using Counter-Harmonic Mean". In: *CoRR* abs/1212.2546 (2012). arXiv: 1212.2546. URL: http://arxiv.org/abs/1212.2546.

– ."A Learning Framework for Morphological Operators using Counter-Harmonic Mean". In: *CoRR* abs/1212.2546 (2012). arXiv: 1212.2546. URL: http://arxiv.org/abs/1212.2546.

Rijsbergen, C. J. van. *Information retrieval*. 2nd ed. London: Butterworths, 1979. URL: http://www.dcs.gla.ac.uk/Keith/Preface.html.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: `1505.04597`. URL: `http://arxiv.org/abs/1505.04597`.

Serra, Jean and Luc Vincent. "An overview of morphological filtering". In: *Circuits, Systems and Signal Processing* 11.1 (Mar. 1992), pp. 47–108. ISSN: 1531-5878. DOI: `10.1007/BF01189221`. URL: `https://doi.org/10.1007/BF01189221`.

Xu, Yongchao, Thierry Géraud, and Isabelle Bloch. "From neonatal to adult brain MR image segmentation in a few seconds using 3D-like fully convolutional network and transfer learning". In: *2017 IEEE International Conference on Image Processing (ICIP)*. Beijing, France: IEEE, Sept. 2017. DOI: 10.1109/ICIP.2017.8297117. URL: https://hal.univ-reims.fr/hal-01735727.

Thanks!

Any questions?

- Tensorboard allows for inspecting layers' weights throughout the training process.

Annex A: Building a Visualization Framework

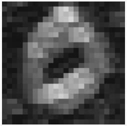
- Tensorboard allows for inspecting layers' weights throughout the training process.
- We needed a way to inspect the output of each layer as well.

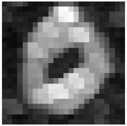
Annex A: Building a Visualization Framework

- Tensorboard allows for inspecting layers' weights throughout the training process.
- We needed a way to inspect the output of each layer as well.
- We built a Keras Callback that saves weights and outputs after every batch.

Annex A: Building a Visualization Framework

- Tensorboard allows for inspecting layers’ weights throughout the training process.
- We needed a way to inspect the output of each layer as well.
- We built a Keras Callback that saves weights and outputs after every batch.
- After training, these weights and outputs are transformed into image sequences:


Init


Batch 64


Batch 96


Batch 160


Batch 512