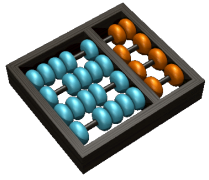
Convolutional Neural Networks

Alexandre Xavier Falcão

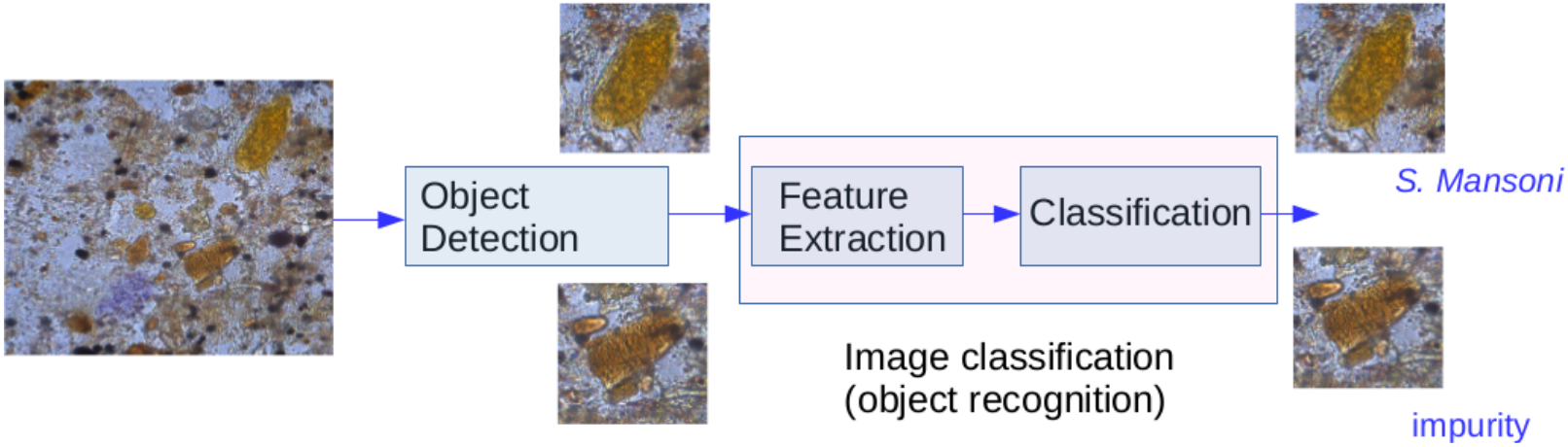
Laboratory of Image Data Science
Institute of Computing — University of Campinas

afalcao@ic.unicamp.br
lids.ic.unicamp.br



Objectives

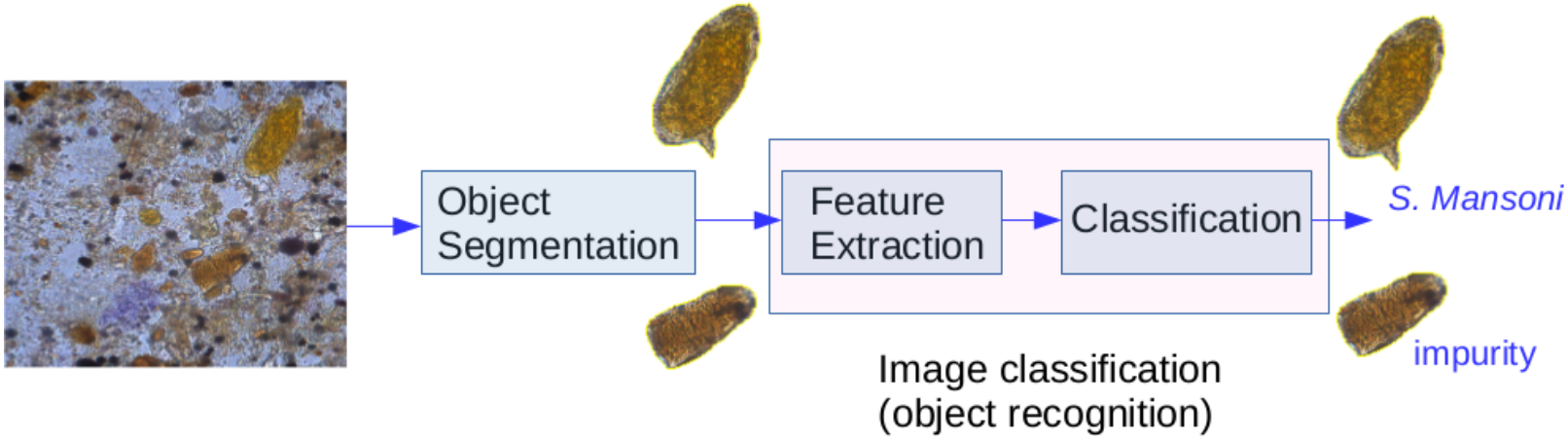
Learn how Convolutional Neural Networks (CNNs) work and how to design CNNs for image classification.



Neural networks may be used in all steps, but it is crucial that their decisions are based on the interest objects rather than on background image features.

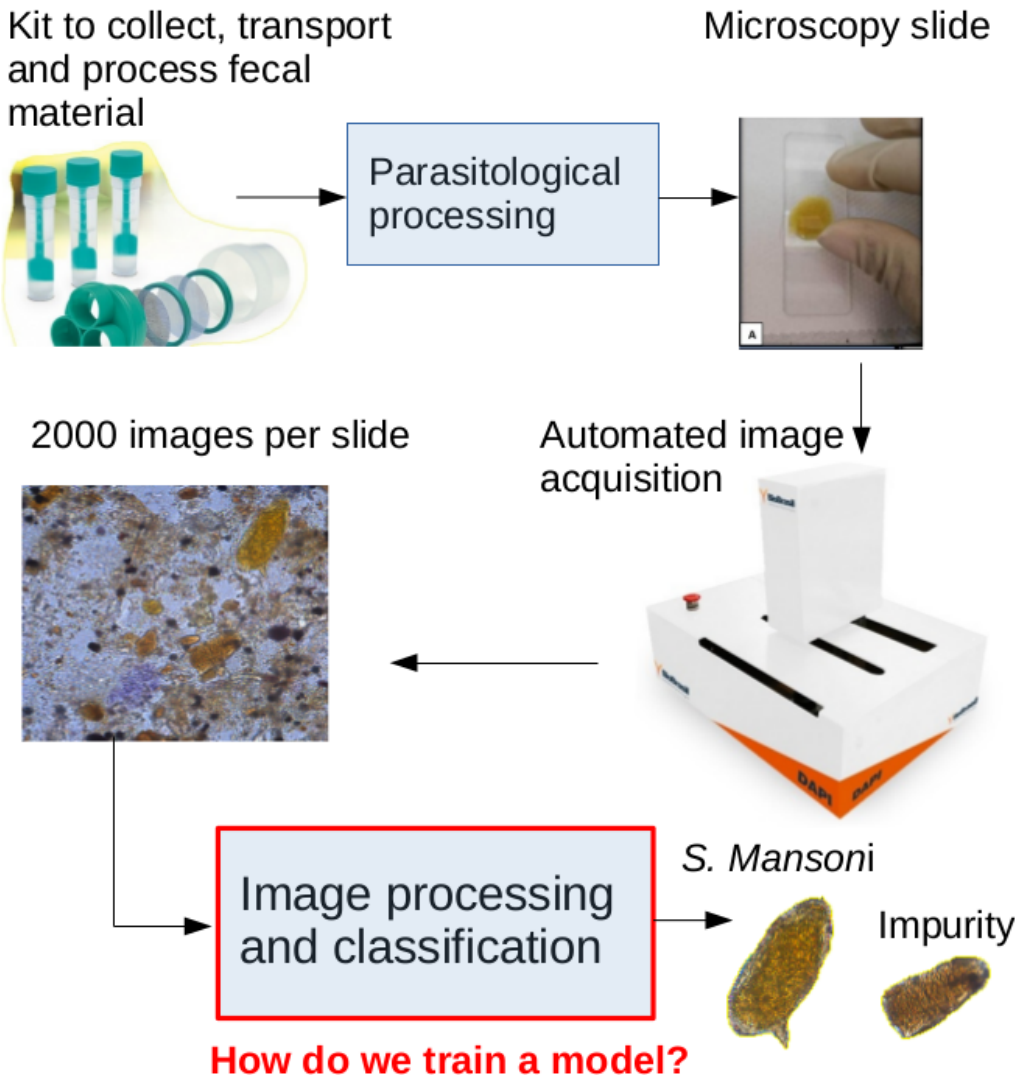
Objectives

Learn how Convolutional Neural Networks (CNNs) work and how to design CNNs for image classification.



Neural networks may be used in all steps, but it is crucial that their decisions are based on the interest objects rather than on background image features.

The real problem related to this example



Automated diagnosis of human intestinal parasites for the 15 most common species in Brazil.

The real problem related to this example

Examples of intestinal parasites (left) and similar impurities (right).

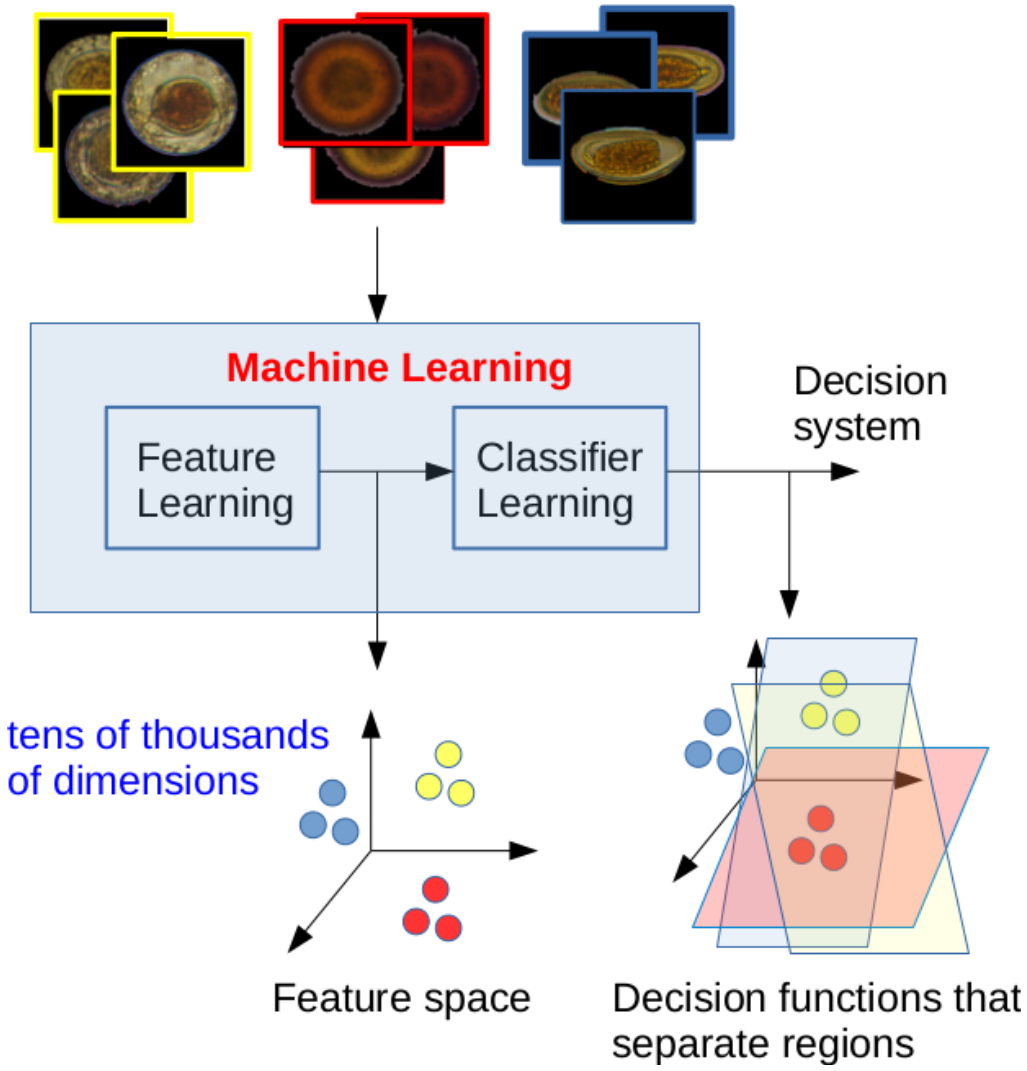
The image classification problem

Assuming that object detection/segmentation has been solved.

For a given training set $\mathcal{I} = \{\hat{I}_j\}_{j=1}^{n}$ of n λ-labeled images from classes $\lambda(\hat{I}_j) \in \{1, 2, \ldots, c\}$, we must find a model that estimates the label $L(\hat{I})$ of any new (test) image $\hat{I}$.

The main challenge in image classification

Images from distinct classes must be mapped into separated groups of points in some multidimensional feature space.

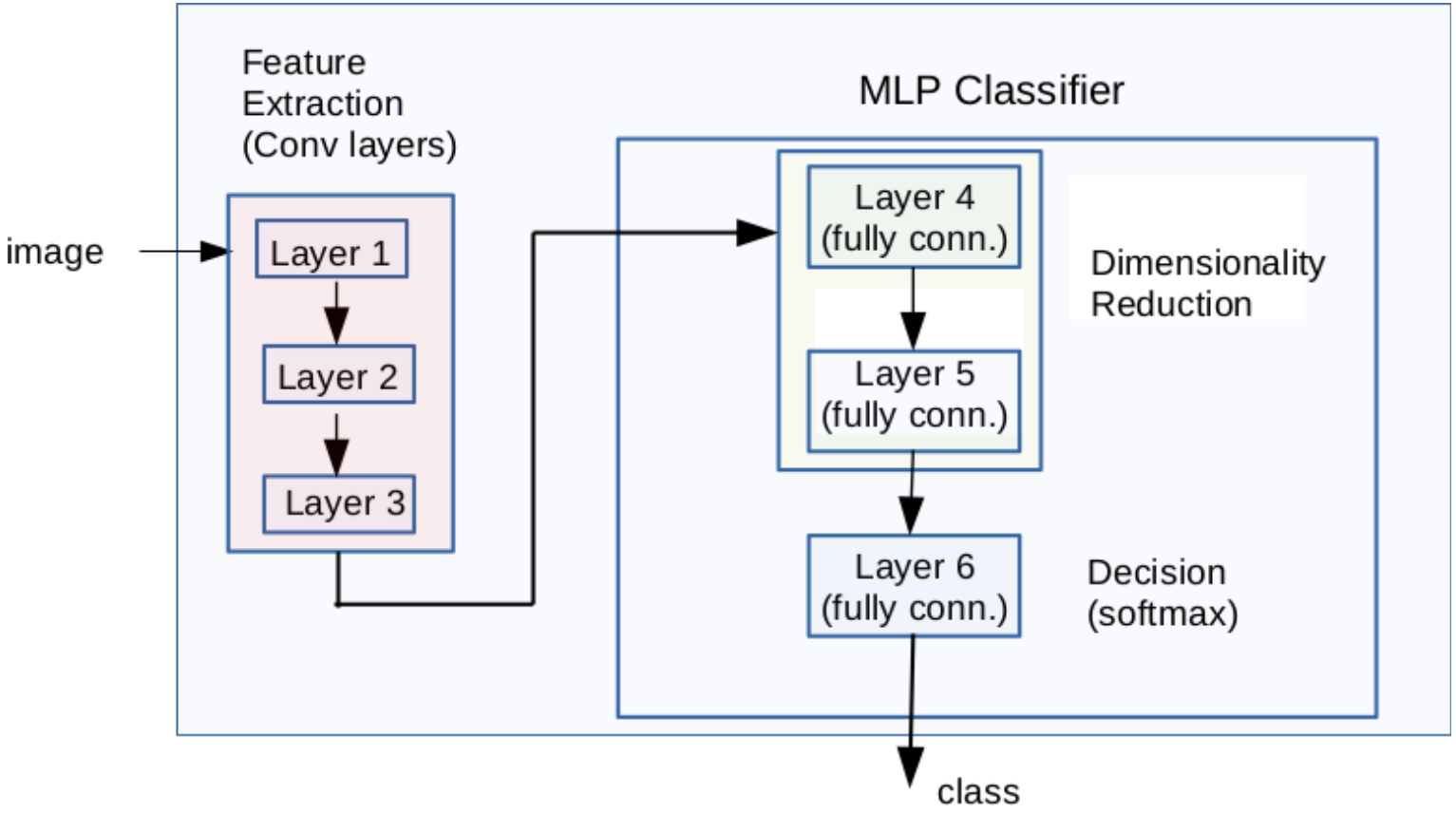


Agenda

- What are CNNs?
- Basic definitions and image processing operations.
- CNNs for image classification — how do they work?
- How to design and test CNNs, and visualize their neuronal activity, using pytorch.
- Hands-on — image classification in pytorch.
- Final remarks.

What are CNNs?

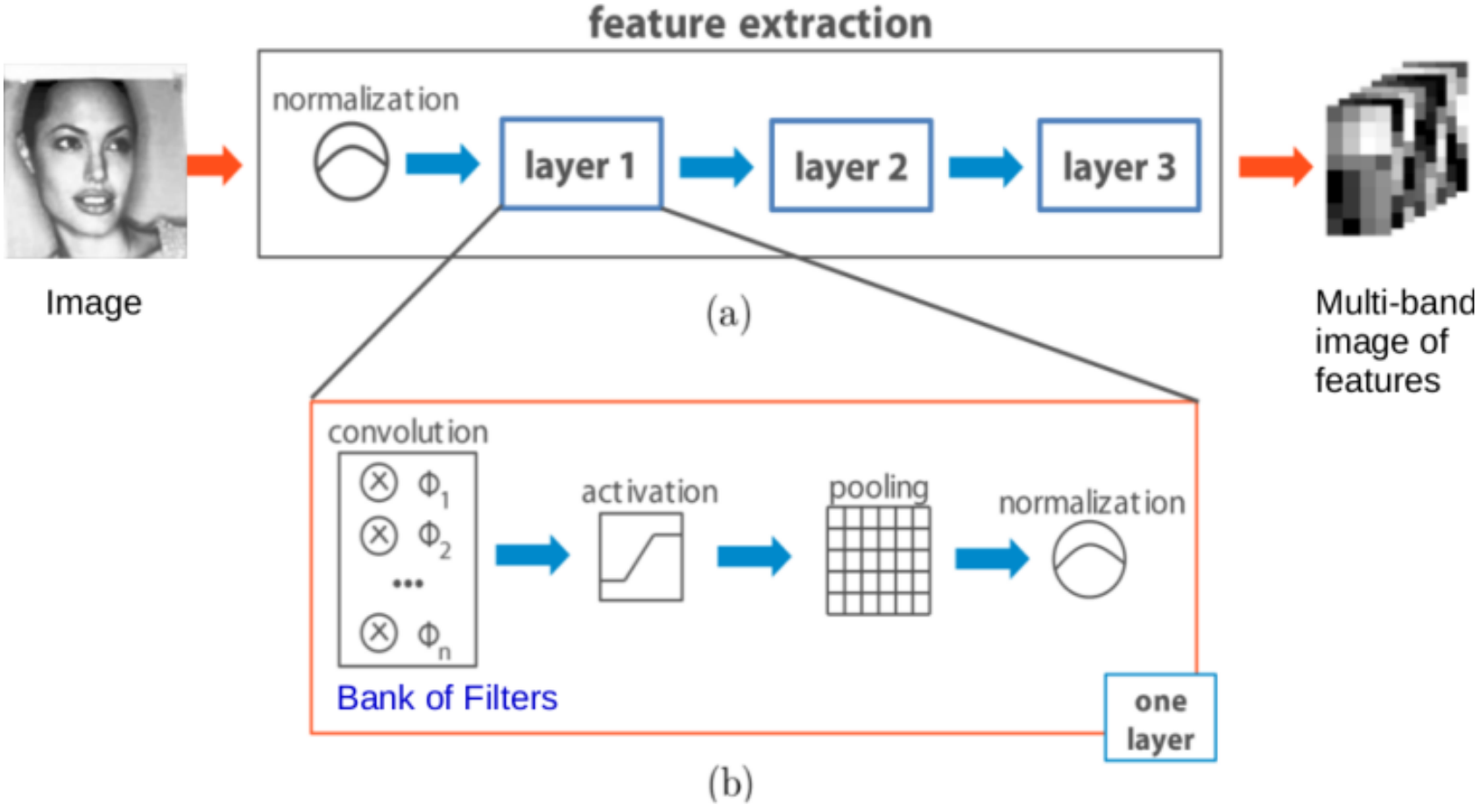
CNNs are artificial neural networks (ANN) with convolutional layers that can extract image features for classification and regression problems.



A possible image classification pipeline.

Convolutional layers

A convolutional layer may consist of four basic operations.



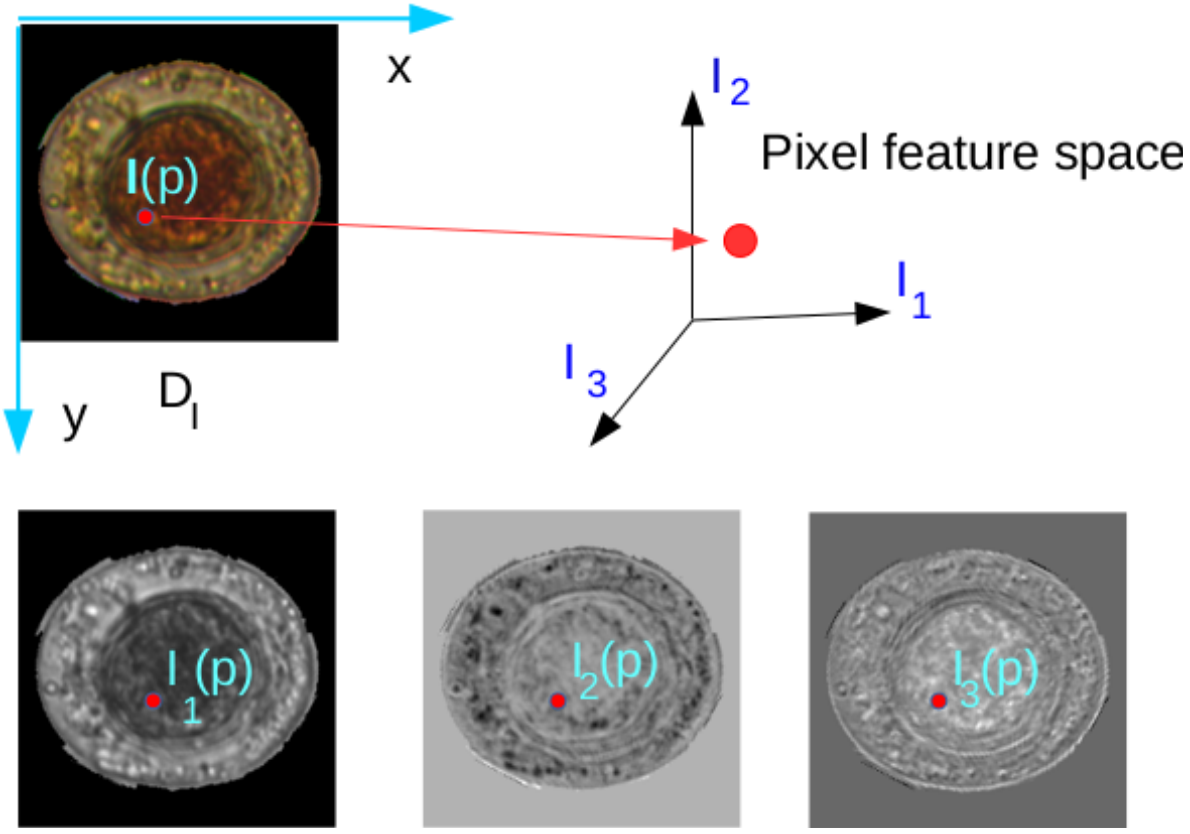
However, there are several types of activation, pooling, and normalization functions.

Basic definitions and image processing operations

- Multiband (multichannel) images and adjacency relations.
- Kernel, kernel bank, and convolution.
- Activation and perceptron (convolution + activation).
- Other image processing operations.
- Global feature spaces and their transformations.

Multiband images

A 2D multiband image $\hat{I}$ is a pair $(D_I, \mathbf{I})$ in which $\mathbf{I}(p) \in \Re^m$ assigns m scalar values to each pixel $p \in D_I \subset \mathcal{Z}^2$.



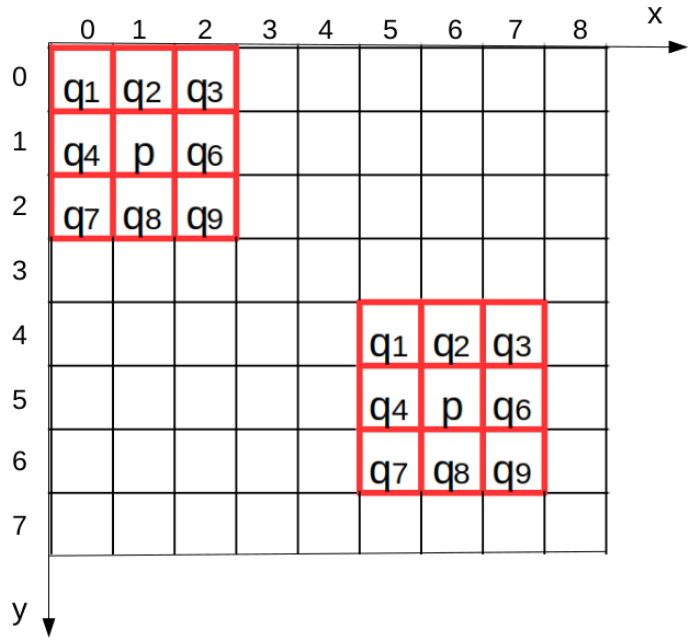
For $m = 3$ bands, each pixel p is represented by a point $\mathbf{I}(p) = (I_1(p), I_2(p), I_3(p)) \in \Re^3$.

Adjacency relation

CNNs usually adopt rectangular adjacency relations A such that

$$A(p) = \{q \in D_I | \quad |x_q - x_p| \leq \frac{w}{2} \quad and \quad |y_q - y_p| \leq \frac{h}{2}\}$$

where $p = (x_p, y_p) \in D_I \subset \mathcal{Z}^2$.



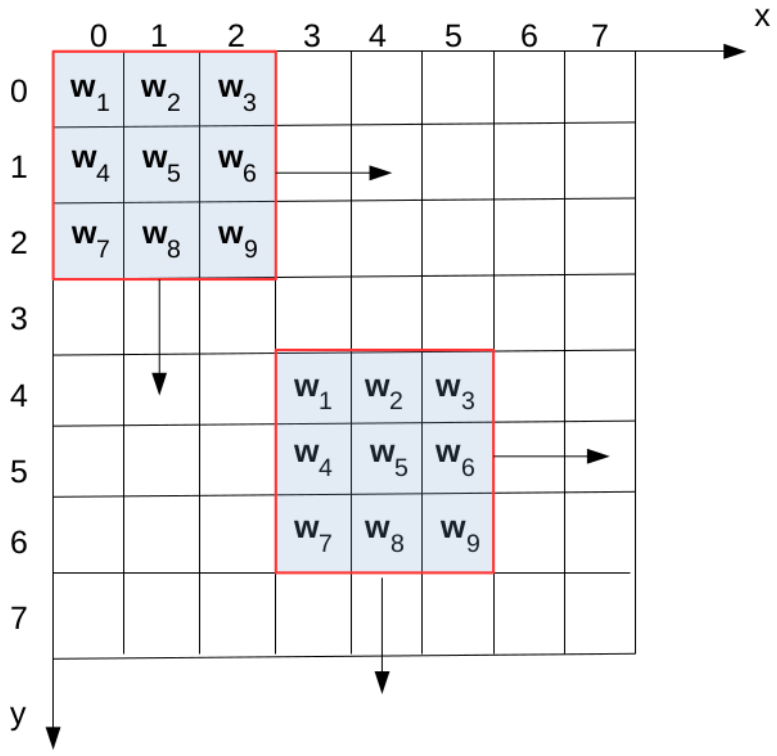
Two examples for pixels p, $p = (1, 1)$ and $p = (6, 5)$, with $w = h = 3$, $A(p) = \{q_1, q_2, \ldots, q_9\}$ where $q_5 = p$.

Basic definitions and image processing operations

- Multiband (multichannel) images and adjacency relations.
- Kernel, kernel bank, and convolution.
- Activation and perceptron (convolution + activation).
- Other image processing operations.
- Global feature spaces and their transformations.

Kernel, kernel bank, and convolution

For an adjacency A, a kernel (filter) $(A, \mathbf{W})$ is a moving image, where $\mathbf{W}(q_i - p)$, $i = 1, 2, \ldots, |A(p)|$, assigns weights $\mathbf{w}_i \in \Re^m$ to the respective adjacent pixels $q_i \in A(p)$ of any $p \in D_I$.



A $3 \times 3 \times m$ filter sliding from top to bottom and left to right over the image domain D_I.

Kernel, kernel bank, and convolution

A kernel bank is just a set of filters and the convolution between an image $\hat{I} = (D_I, \mathbf{I})$ and a filter $(A, \mathbf{W})$ creates a grayscale image $\hat{J} = (D_J, J)$,

$$\begin{aligned} J(p) &= \sum_{i=1}^{|A(p)|} \langle \mathbf{I}(q_i), \mathbf{w}_i \rangle, \\ \langle \mathbf{I}(q_i), \mathbf{w}_i \rangle &= \sum_{k=1}^{m} I_k(q_i) w_{i,k}, \end{aligned}$$

for all $p \in D_I$, with D_J forced to be $\subseteq D_I$.

Convolution

The convolution between a grayscale image $\hat{I} = (D_I, I)$ and a $3 \times 3 \times 1$ filter generating a grayscale image $\hat{J} = (D_I, J)$.

Image function I

	3	2	2	5	
	2	2	5	1	
	2	3	5	0	
	0	4	4	1	

Zero-padding in red

Image function J

6	1	7	-9
9	8	-2	-17
12	13	-10	-19
11	11	-9	-13

same image domain

Kernel 3 x 3 x 1

-1	0	1
-2	0	2
-1	0	1

Convolution

The convolution between an image $\hat{I} = (D_I, \mathbf{I})$ and a bank of b kernels $\{(A, \mathbf{W}_k)\}_{k=1}^{b}$ will produce an image $\hat{J} = (D_I, \mathbf{J})$ with b bands J_k, $k \in [1, b]$,

$$J_k(p) = \sum_{i=1}^{|A(p)|} \langle \mathbf{I}(q_i), \mathbf{w}_{k,i} \rangle.$$

Convolution

The convolution between an image $\hat{I} = (D_I, \mathbf{I})$ and a bank of b kernels $\{(A, \mathbf{W}_k)\}_{k=1}^{b}$ will produce an image $\hat{J} = (D_I, \mathbf{J})$ with b bands J_k, $k \in [1, b]$,

$$J_k(p) = \sum_{i=1}^{|A(p)|} \langle \mathbf{I}(q_i), \mathbf{w}_{k,i} \rangle.$$

The adjacency A is usually fixed for all kernels because the convolution can be very efficiently computed by matrix multiplication using parallel programming.

Convolution

The convolution between an image $\hat{I} = (D_I, \mathbf{I})$ and a bank of b kernels $\{(A, \mathbf{W}_k)\}_{k=1}^{b}$ will produce an image $\hat{J} = (D_I, \mathbf{J})$ with b bands J_k, $k \in [1, b]$,

$$J_k(p) = \sum_{i=1}^{|A(p)|} \langle \mathbf{I}(q_i), \mathbf{w}_{k,i} \rangle.$$

The adjacency A is usually fixed for all kernels because the convolution can be very efficiently computed by matrix multiplication using parallel programming.

One can also reduce the number of bands from m to $b < m$ by convolving an image with b filters of size $1 \times 1 \times m$.

Basic definitions and image processing operations

- Multiband (multichannel) images and adjacency relations.
- Kernel, kernel bank, and convolution.
- Activation and perceptron (convolution + activation).
- Other image processing operations.
- Global feature spaces and their transformations.

Activation

Among several activation functions, we will adopt the Rectified Linear Unit (ReLU).

Activation

Among several activation functions, we will adopt the Rectified Linear Unit (ReLU).

From the output $\hat{J} = (D_I, \mathbf{J})$ of a convolution, ReLU creates an image $\hat{R} = (D_I, \mathbf{R})$, $\mathbf{R}(p) = (R_1(p), R_2(p), \ldots, R_b(p))$,

$$R_k(p) \quad = \quad \max\{0, J_k(p)\},$$

for $p \in D_I$ and $k \in [1, b]$.

Activation

Among several activation functions, we will adopt the Rectified Linear Unit (ReLU).

From the output $\hat{J} = (D_I, \mathbf{J})$ of a convolution, ReLU creates an image $\hat{R} = (D_I, \mathbf{R})$, $\mathbf{R}(p) = (R_1(p), R_2(p), \ldots, R_b(p))$,

$$R_k(p) \quad = \quad \max\{0, J_k(p)\},$$

for $p \in D_I$ and $k \in [1, b]$.

Image function J

6	1	7	-9
9	8	-2	-17
12	13	-10	-19
11	11	-9	-13

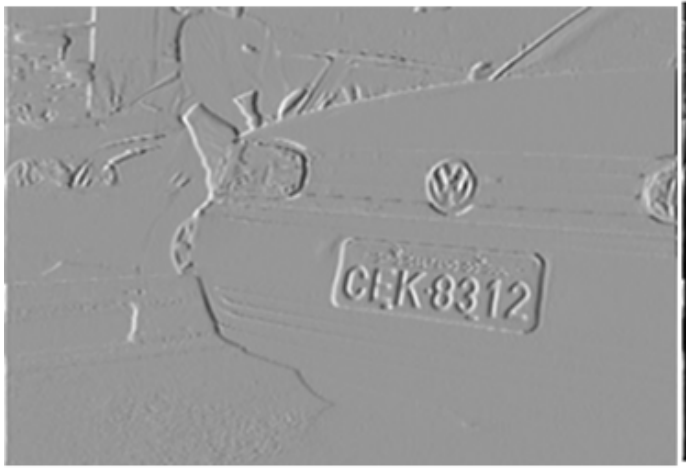
Image function R

6	1	7	0
9	8	0	0
12	13	0	0
11	11	0	0

Convolution followed by activation

Kernel 3 x 3 x 1

-1	0	1
-2	0	2
-1	0	1



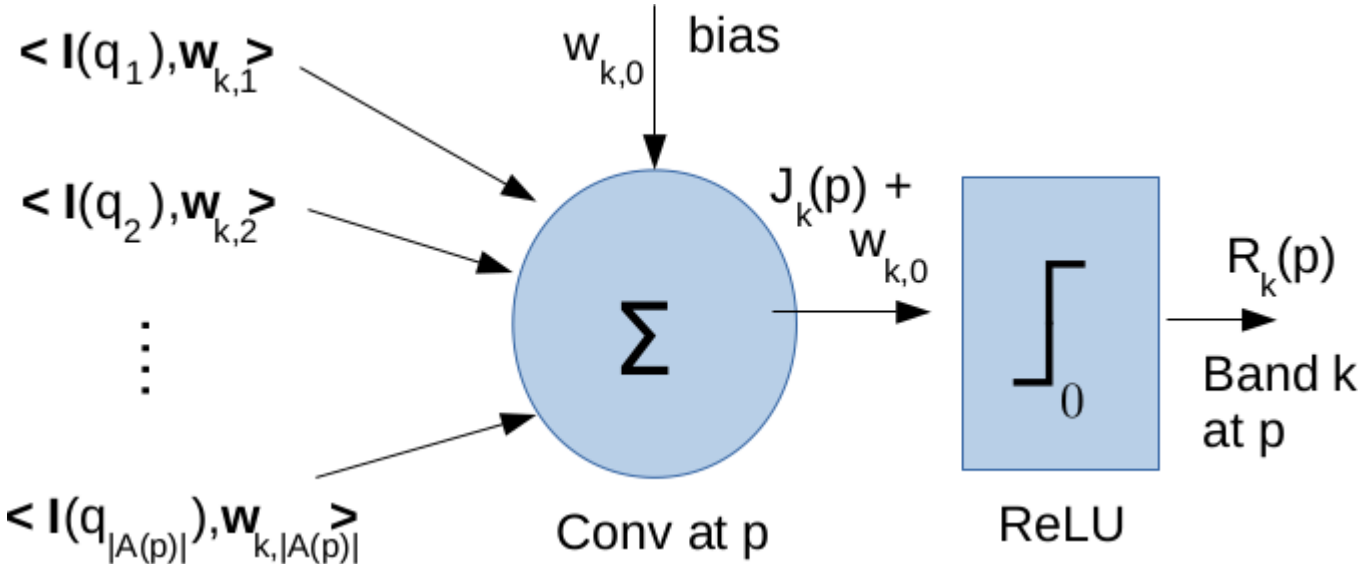
After convolution

After ReLU

Transitions from dark to bright are enhanced.

Convolution, bias, and activation

By convolving $\hat{I} = (D_I, \mathbf{I})$ and the k-th filter $\{(A, \mathbf{W}_k)\}$, $k \in [1, b]$, adding a bias $w_{k,0} \in \Re$ to each output $J_k(p)$, and applying a ReLU operation, we have one perceptron per pixel $p \in D_I$ (artificial neuron).



$$J_k(p) = \sum_{i=1}^{|A(p)|} \langle \mathbf{I}(q_i), \mathbf{w}_{k,i} \rangle$$

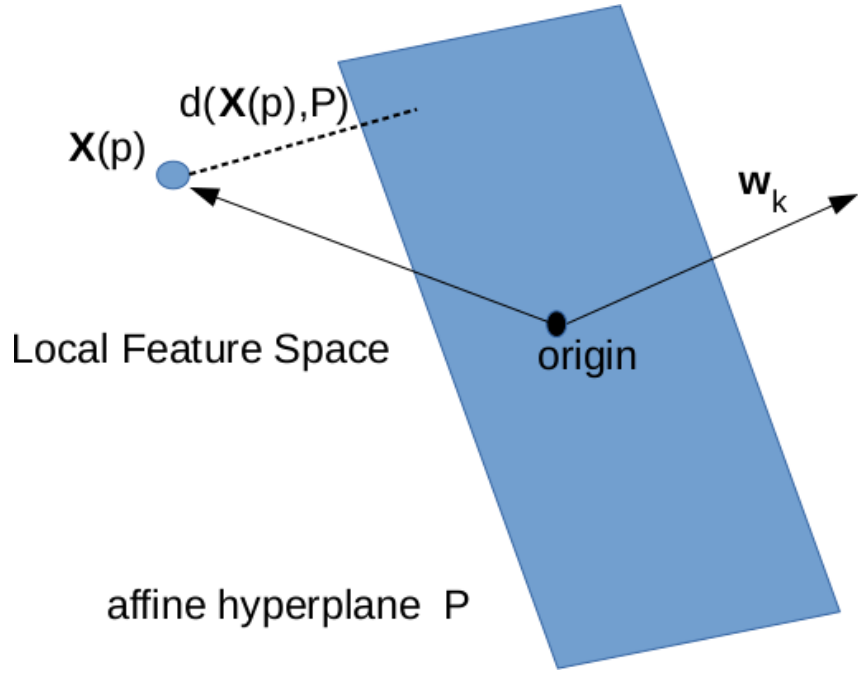
$$R_k(p) = \max\{0, J_k(p) + w_{k,0}\}$$

Convolution, bias, and activation

Let $\mathbf{X}(p) \in \Re^{|A(p)| \times m}$ be a local feature vector

$$\mathbf{X}(p) \quad = \quad (\mathbf{I}(q_1), \mathbf{I}(q_2), \ldots, \mathbf{I}(q_{|A(p)|}))$$

and P be an affine hyperplane $\langle x, \mathbf{w}_k \rangle + w_{k,0} = 0$ in $\Re^{|A(p)| \times m}$.



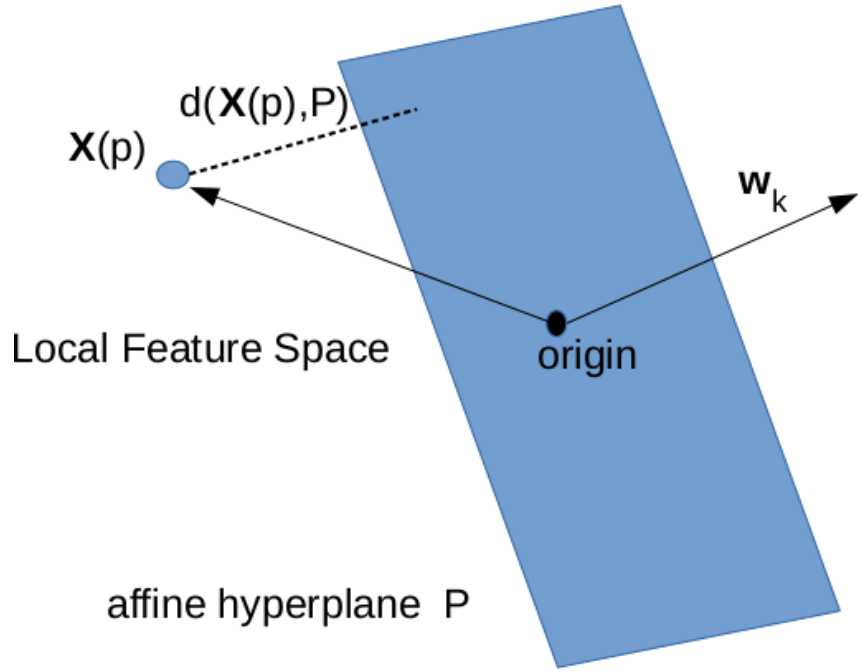
The distance $d(\mathbf{X}(p), P)$ from $\mathbf{X}(p)$ to the hyperplane is given by

$$d(\mathbf{X}(p), P) \quad = \quad \frac{\langle \mathbf{X}(p), \mathbf{w}_k \rangle + w_{k,0}}{\|\mathbf{w}_k\|} = \frac{J_k(p) + w_{k,0}}{\|\mathbf{w}_k\|}$$

Convolution, bias, and activation

The perceptron at p selects $R_k(p)$ as a local feature only when the activation

$$\langle \mathbf{X}(p), \mathbf{w}_k \rangle + w_{k,0} \quad = \quad J_k(p) + w_{k,0} > 0,$$



meaning that, the bias moves P such that $\mathbf{X}(p)$ falls in its positive side.

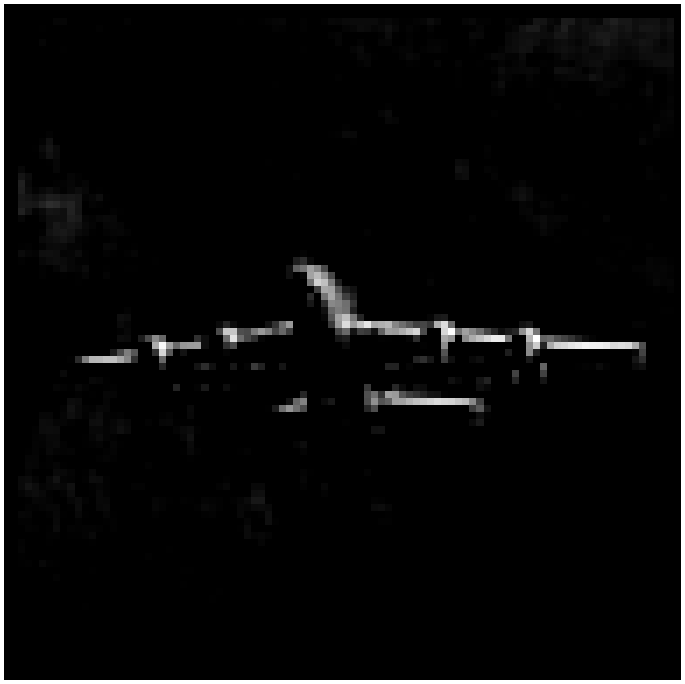
Convolution, bias, and activation

Therefore, the convolution, bias, and activation — a neuronal layer (layer of perceptrons $p \in D_I$) — should extract and select pixel features in parts that best represent the object characteristics from each class of the image classification problem.

Output of activation for four random kernels: some kernels may be better than others and some may be redundants.

Convolution, bias, and activation

Therefore, the convolution, bias, and activation — a neuronal layer (layer of perceptrons $p \in D_I$) — should extract and select pixel features in parts that best represent the object characteristics from each class of the image classification problem.



Output of activation for four random kernels: some kernels may be better than others and some may be redundants.

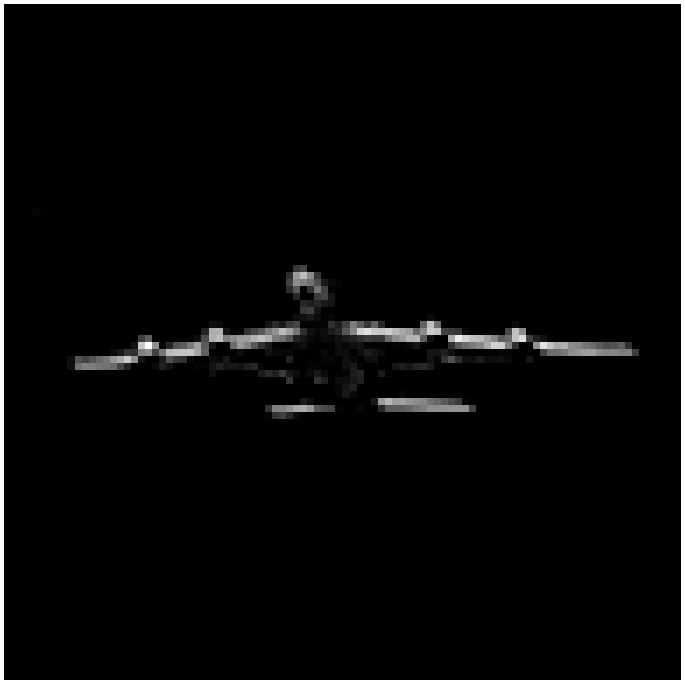
Convolution, bias, and activation

Therefore, the convolution, bias, and activation — a neuronal layer (layer of perceptrons $p \in D_I$) — should extract and select pixel features in parts that best represent the object characteristics from each class of the image classification problem.

Output of activation for four random kernels: some kernels may be better than others and some may be redundants.

Convolution, bias, and activation

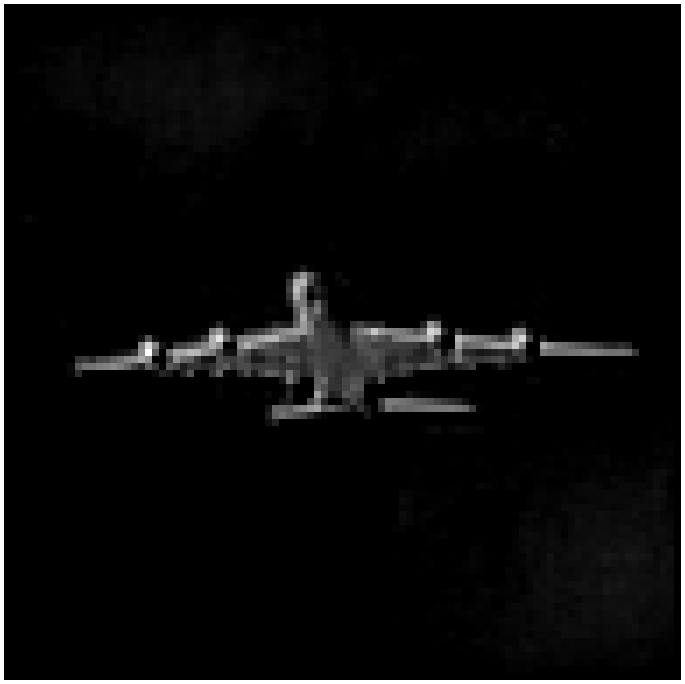
Therefore, the convolution, bias, and activation — a neuronal layer (layer of perceptrons $p \in D_I$) — should extract and select pixel features in parts that best represent the object characteristics from each class of the image classification problem.



Output of activation for four random kernels: some kernels may be better than others and some may be redundants.

Convolution, bias, and activation

Therefore, the convolution, bias, and activation — a neuronal layer (layer of perceptrons $p \in D_I$) — should extract and select pixel features in parts that best represent the object characteristics from each class of the image classification problem.



Output of activation for four random kernels: some kernels may be better than others and some may be redundants.

Basic definitions and image processing operations

- Multiband (multichannel) images and adjacency relations.
- Kernel, kernel bank, and convolution.
- Activation and perceptron (convolution + activation).
- Other image processing operations.
- Global feature spaces and their transformations.

Pooling

The activations $R_k(p)$ related to an object of interest might also appear at nearby positions within and across images.

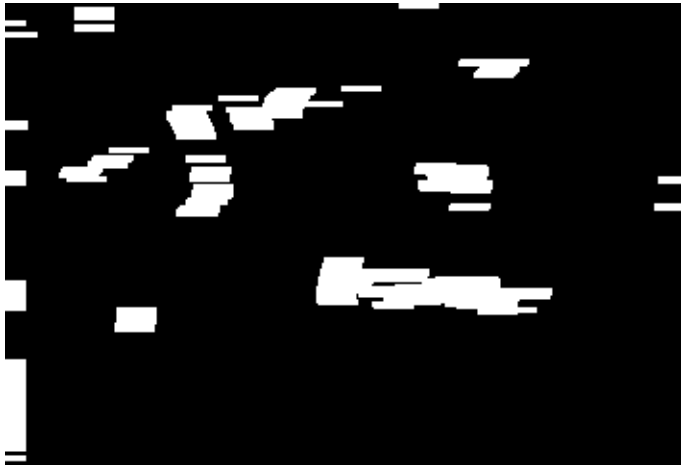
Pooling

The activations $R_k(p)$ related to an object of interest might also appear at nearby positions within and across images.

Max-pooling can aggregate them by transforming $\hat{R} = (D_I, \mathbf{R})$ into $\hat{P} = (D_I, \mathbf{P})$, $\mathbf{P}(p) = (P_1(p), P_2(p), \ldots, P_b(p))$,

$$P_k(p) = \max_{\forall q \in B(p)} \{R_k(q)\},$$

where B is an adjacency relation.



In this case, the widest component is the plate.

Pooling

The activations $R_k(p)$ related to an object of interest might also appear at nearby positions within and across images.

Max-pooling can aggregate them by transforming $\hat{R} = (D_I, \mathbf{R})$ into $\hat{P} = (D_I, \mathbf{P})$, $\mathbf{P}(p) = (P_1(p), P_2(p), \ldots, P_b(p))$,

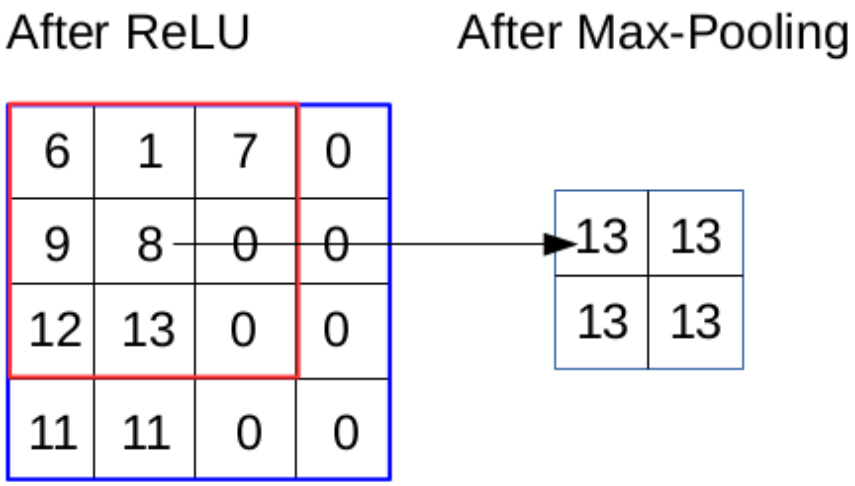
$$P_k(p) = \max_{\forall q \in B(p)} \{R_k(q)\},$$

where B is an adjacency relation.

In this case, the widest component is the plate.

Pooling with stride

It is also common to down-sampling the input image with displacements $s_x \geq 1$ and $s_y \geq 1$, called strides.



For a stride $s_x = s_y = 2$ and a 3×3 adjacency relation B, the image domain D_P of $\hat{P}$ will be $\frac{D_I}{2 \times 2}$.

Pooling

Other examples that create $\hat{P} = (D_I, \mathbf{P})$ by pooling are min-pooling and average pooling.

- Min-pooling:

$$P_k(p) = \min_{\forall q \in B(p)} \{R_k(q)\}.$$

- Average pooling:

$$P_k(p) = \frac{1}{|B(p)|} \sum_{\forall q \in B(p)} \{R_k(q)\}.$$

Indeed, any other image filtering could be used here to eliminate undesirable features and/or aggregate the desirable ones for better image classification.

Normalization

Normalizations may be applied to any image $\hat{I} = (D_I, \mathbf{I})$ with m bands or, in batch, to a set $\mathcal{I} = \{\hat{I}_j\}_{j=1}^{n}$ of m-band images before/after any step in a convolutional layer.

They are important to avoid discrepancies among local features along the network.

They create a new image $\hat{N} = (D_I, \mathbf{N})$ with m bands or a new set $\mathcal{N} = \{\hat{N}_j\}_{j=1}^{n}$ of m-band images.

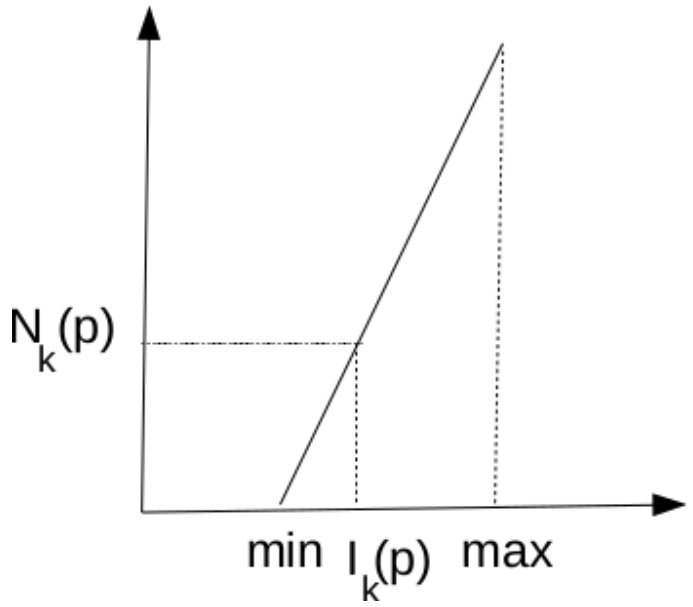
Linear normalization

For $\hat{N} = (D_I, \mathbf{N})$, $\mathbf{N}(p) = (N_1(p), N_2(p), \ldots, N_m(p))$,

$$N_k(p) = \frac{I_k(p) - \min_{\forall q \in D_I}\{I_k(q)\}}{\max_{\forall q \in D_I}\{I_k(q)\} - \min_{\forall q \in D_I}\{I_k(q)\}},$$

$$N_k(p) = \frac{I_k(p) - \min_{j=1}^{n}\{I_{j,k}(p)\}}{\max_{j=1}^{n}\{I_{j,k}(p)\} - \min_{j=1}^{n}\{I_{j,k}(p)\}},$$

$k \in [1, m]$ and $p \in D_I$, we have a linear normalization.

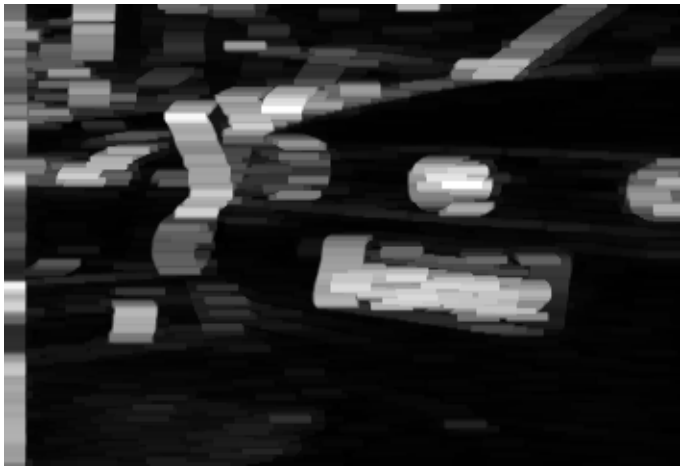


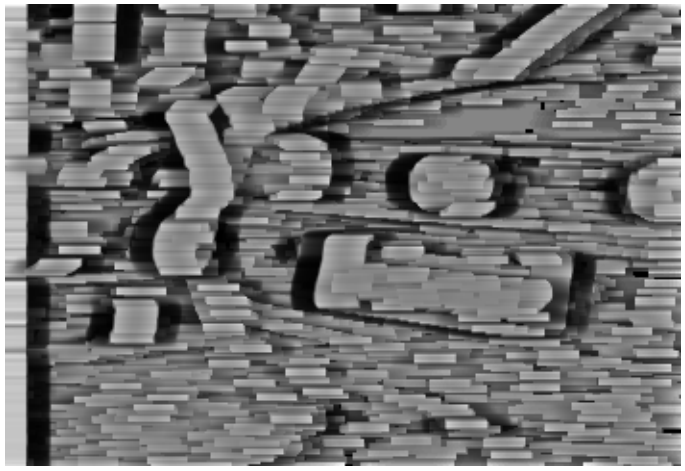
Divisive normalization

Divisive normalization can enhance subtle and isolated activations within an adjacency C, creating $\hat{N} = (D_I, \mathbf{N})$, with

$$N_k(p) = \frac{I_k(p)}{\sqrt{\sum_{\forall q \in C(p)} I_k^2(q)}},$$

for $k \in [1, m]$ and $p \in D_I$.





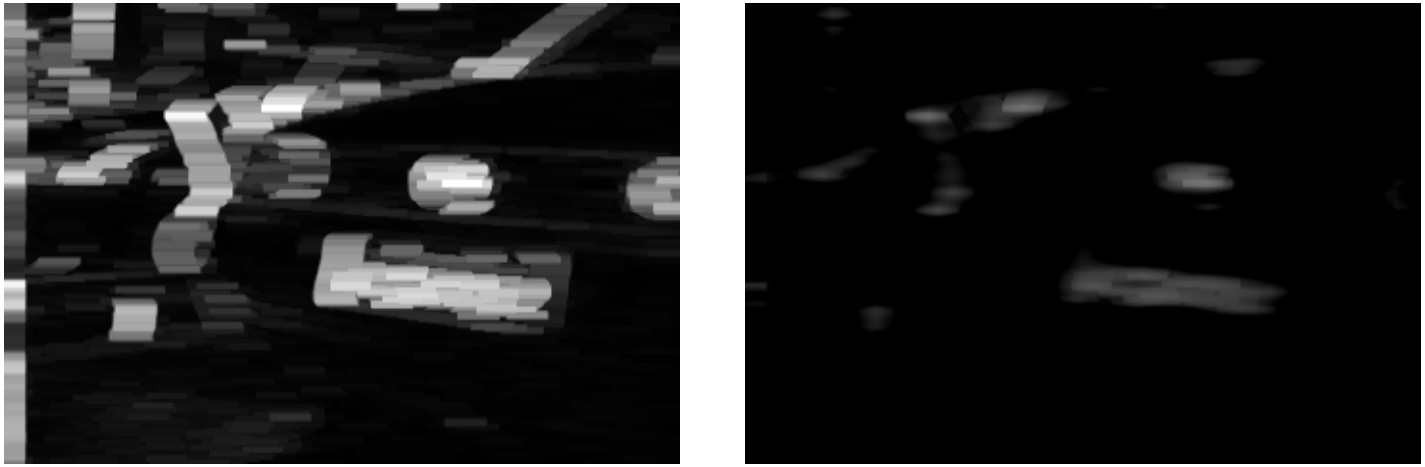
C is rectangular with $w = 25$ and $h = 5$. The normalized image before and after residue+ReLU (right).

Divisive normalization

Divisive normalization can enhance subtle and isolated activations within an adjacency C, creating $\hat{N} = (D_I, \mathbf{N})$, with

$$N_k(p) = \frac{I_k(p)}{\sqrt{\sum_{\forall q \in C(p)} I_k^2(q)}},$$

for $k \in [1, m]$ and $p \in D_I$.



C is rectangular with $w = 25$ and $h = 5$. The normalized image before and after residue+ReLU (right).

Batch normalization

Batch normalization is very useful to standardize local features and eliminate the need of bias learning.

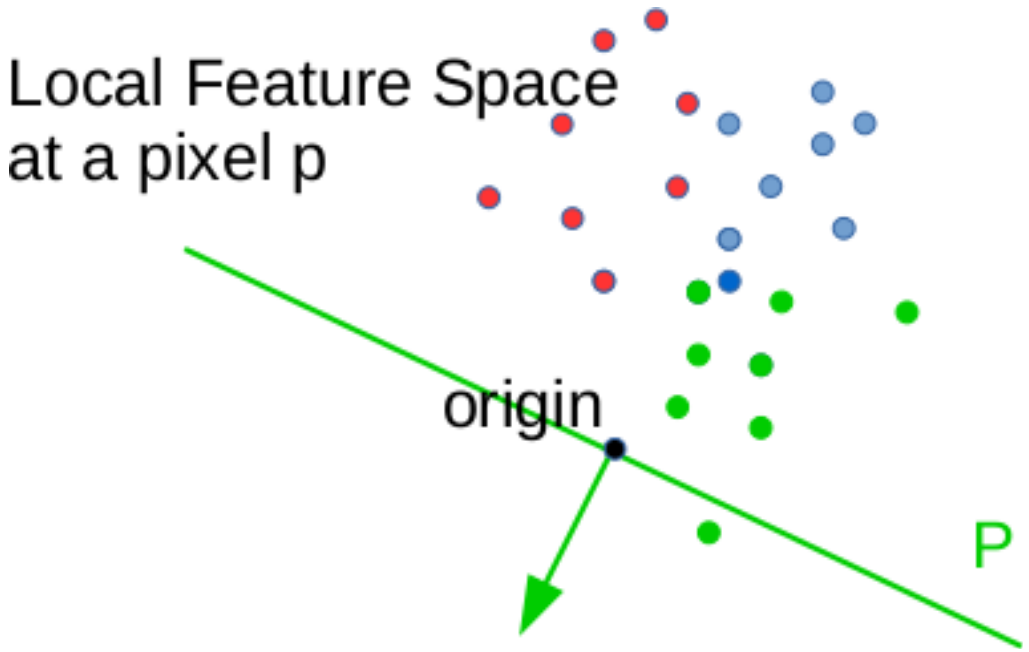
It creates an image $\hat{N} = (D_I, \mathbf{N})$, with

$$
\begin{aligned}
N_k(p) &= \frac{I_k(p) - \mu_k(p)}{\sigma_k(p)}\gamma + \beta, \\
\mu_k(p) &= \frac{1}{n}\sum_{j=1}^{n} I_{j,k}(p), \\
\sigma_k^2(p) &= \frac{1}{n-1}\sum_{j=1}^{n}(I_k(p) - \mu_k(p))^2,
\end{aligned}
$$

for $k \in [1, m]$, $p \in D_I$, and $\gamma, \beta \in \Re$ are parameters that can be learned and even undo this operation. Let $\gamma = 1$ and $\beta = 0$ be their default values.

Batch normalization

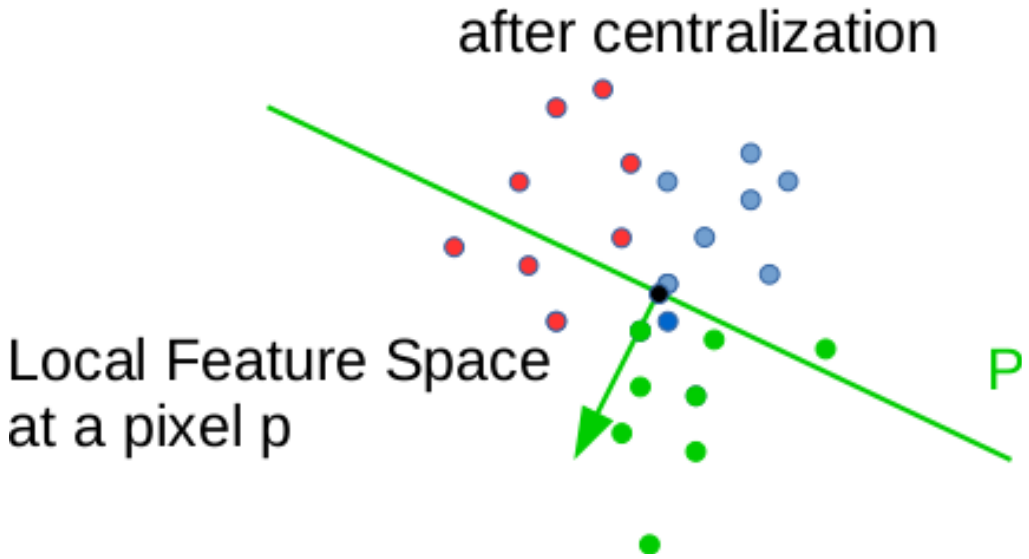
Batch normalization affects the local feature space with points $\mathbf{X}_j(p)$ from an image set $\mathcal{I} = \{\hat{I}_j\}_{j=1}^{n}$ for all pixels $p \in D_I$.



Just the centralization of the point cloud already shows that training can adjust a kernel to select more features from a given class with no need of bias.

Batch normalization

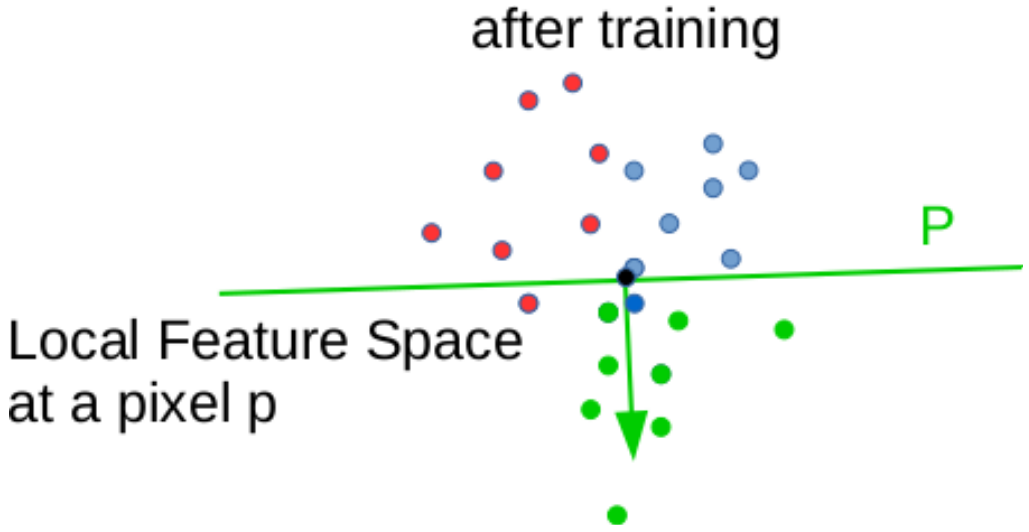
Batch normalization affects the local feature space with points $\mathbf{X}_j(p)$ from an image set $\mathcal{I} = \{\hat{I}_j\}_{j=1}^{n}$ for all pixels $p \in D_I$.



Just the centralization of the point cloud already shows that training can adjust a kernel to select more features from a given class with no need of bias.

Batch normalization

Batch normalization affects the local feature space with points $\mathbf{X}_j(p)$ from an image set $\mathcal{I} = \{\hat{I}_j\}_{j=1}^{n}$ for all pixels $p \in D_I$.



Just the centralization of the point cloud already shows that training can adjust a kernel to select more features from a given class with no need of bias.

Basic definitions and image processing operations

- Multiband (multichannel) images and adjacency relations.
- Kernel, kernel bank, and convolution.
- Activation and perceptron (convolution + activation).
- Other image processing operations.
- Global feature spaces and their transformations.

Global feature spaces and their transformations

- One may create a global feature vector for image classification at any step of a convolutional layer.

Global feature spaces and their transformations

- One may create a global feature vector for image classification at any step of a convolutional layer.

- For example, at the input layer,

$$\mathbf{X}(\hat{I}) \quad = \quad \mathbf{B}_1 \frown \mathbf{B}_2 \frown \ldots \frown \mathbf{B}_m \in \Re^{|D_I| \times m}$$

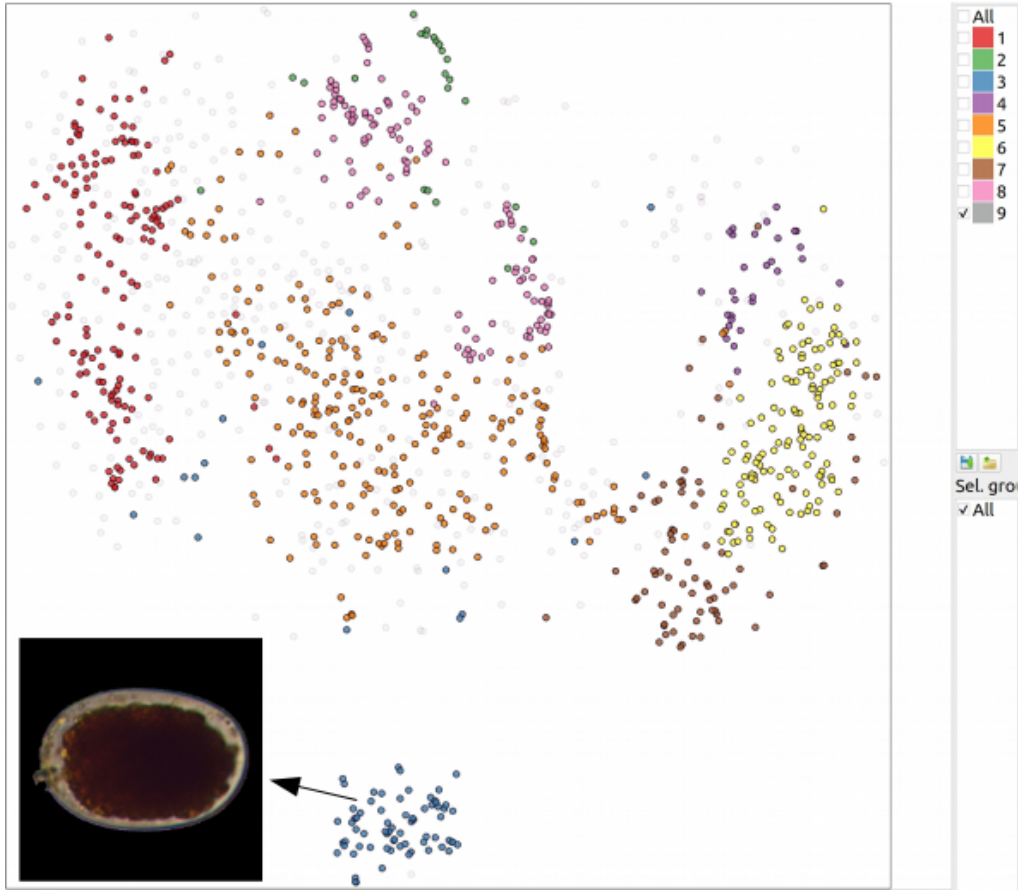
results from the concatenation of each band vector $\mathbf{B}_k$ of $\hat{I}$, $k \in [1, m]$, where

$$\mathbf{B}_k \quad = \quad (I_k(p_1), I_k(p_2), \ldots, I_k(p_{|D_I|})) \in \Re^{|D_I|}.$$

This operation is known as flattening.

Global feature space

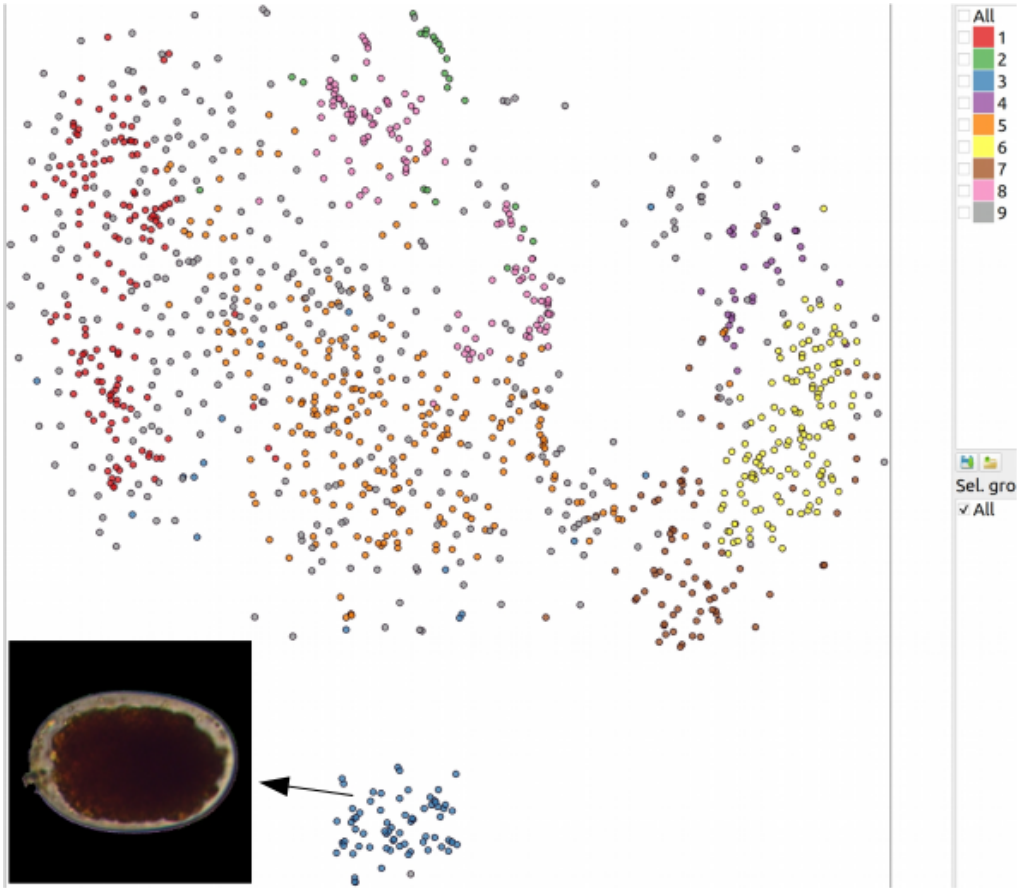
For a given set of images, one can use non-linear projections to visualize the distribution of points in $\Re^{|D_I|}$ formed by the global feature vectors of those images.



Feature projection (t-SNE) of helminth eggs at the input layer w/o impurities.

Global feature space

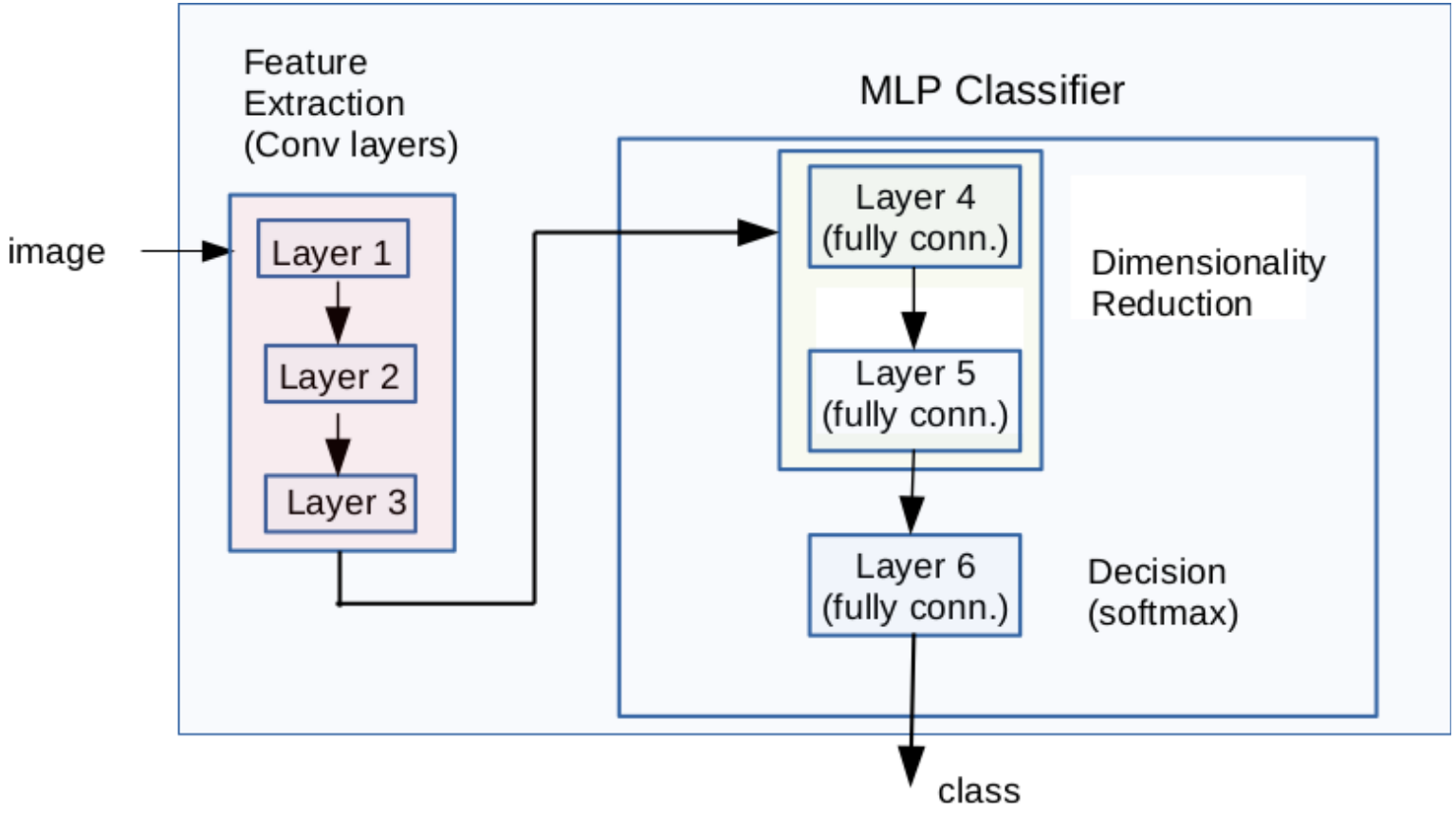
For a given set of images, one can use non-linear projections to visualize the distribution of points in $\Re^{|D_I|}$ formed by the global feature vectors of those images.



Feature projection (t-SNE) of helminth eggs at the input layer w/o impurities.

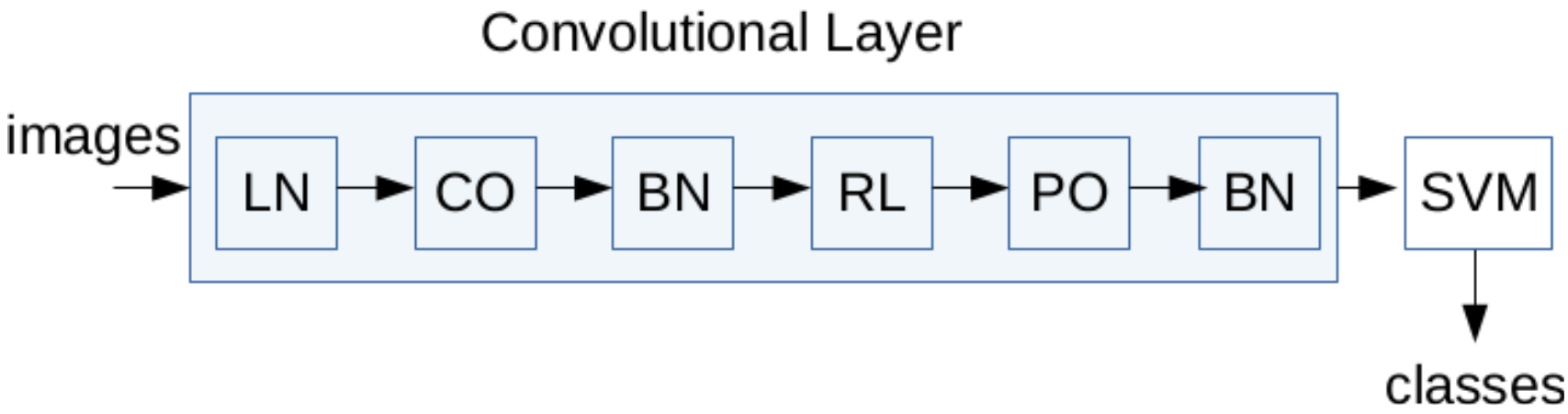
Global feature space transformations

A CNN essentially transforms the global feature space of the input images into new spaces along the convolutional and fully-connected layers.



Global feature space transformations

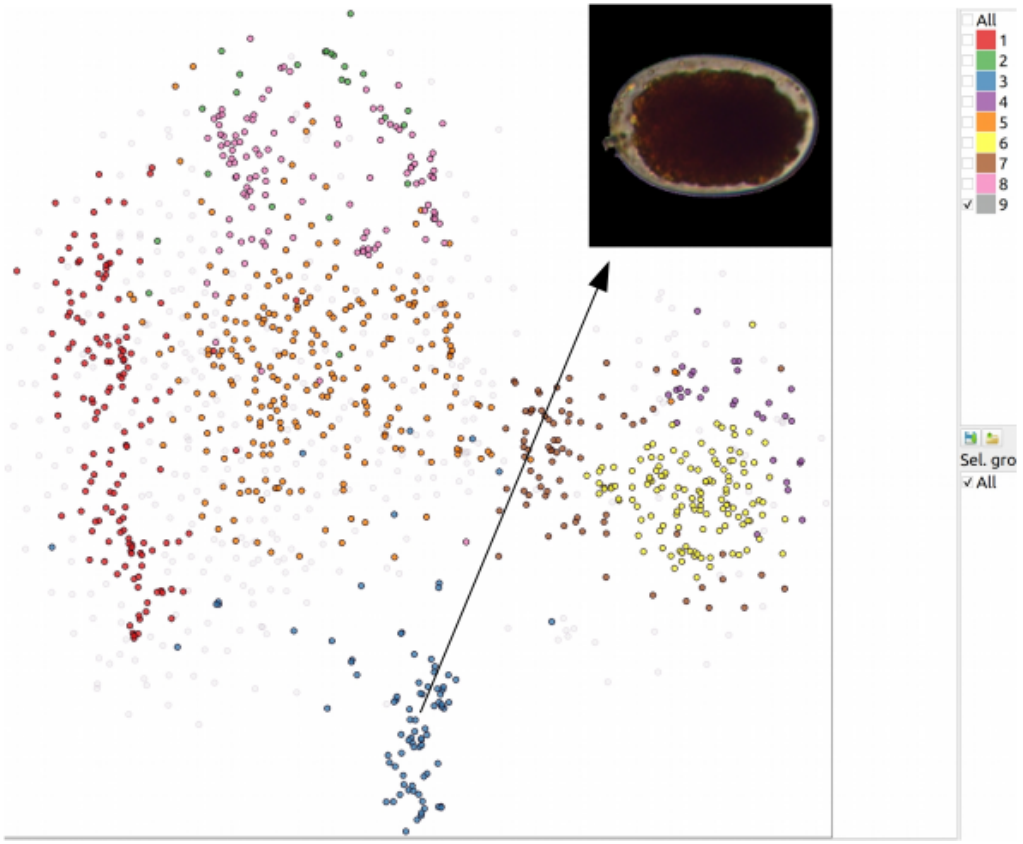
Even inside each convolutional layer, there are global feature space transformations.



LN - linear normalization, CO - convolution, BN - batch normalization, RL - ReLU, PO - pooling, and SVM - support vector machine.

Global feature space transformations

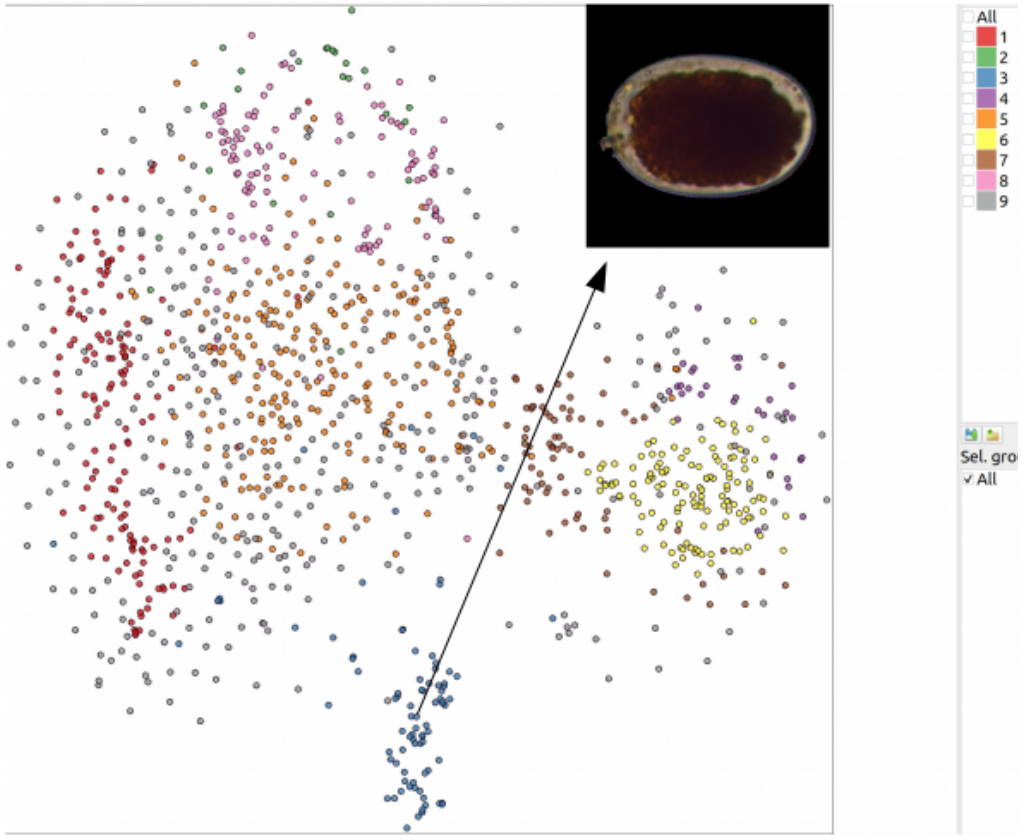
After applying the previous pipeline using a random kernel bank with $5 \times 5 \times 3$ filters, SVM can improve about 3% its performance on unseen test sets.



Feature projection (t-SNE) of helminth eggs at the output layer w/o impurities.

Global feature space transformations

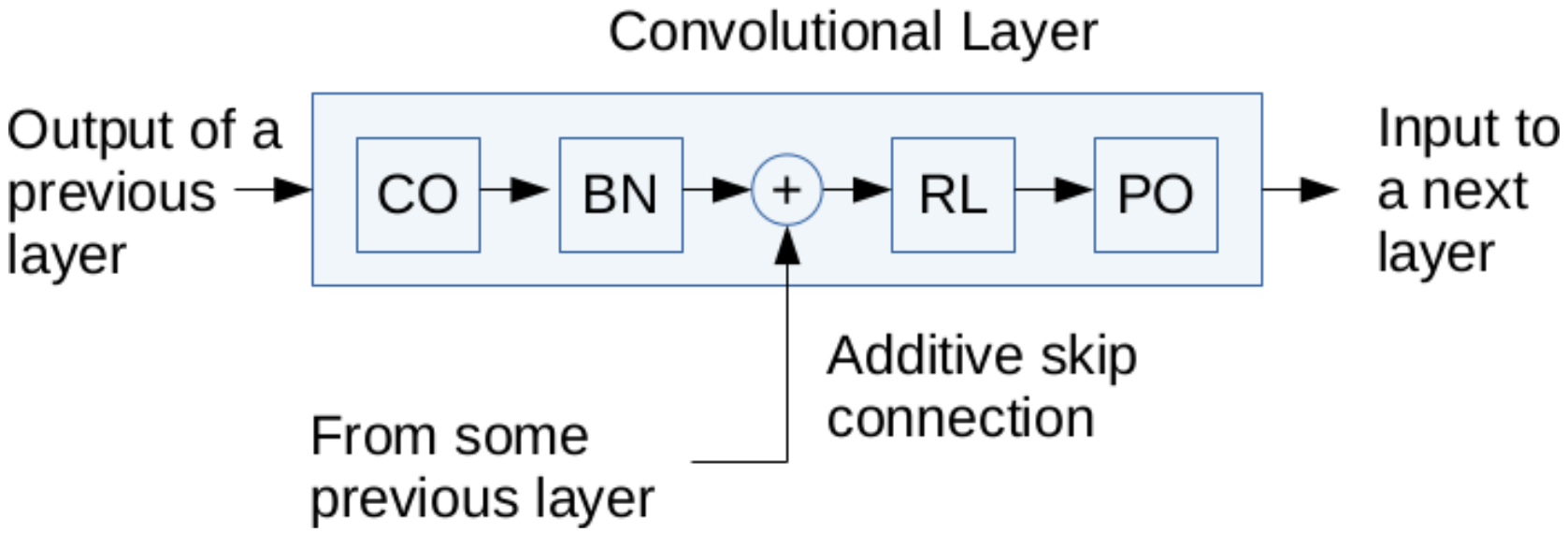
After applying the previous pipeline using a random kernel bank with $5 \times 5 \times 3$ filters, SVM can improve about 3% its performance on unseen test sets.



Feature projection (t-SNE) of helminth eggs at the output layer w/o impurities.

Global feature space transformations

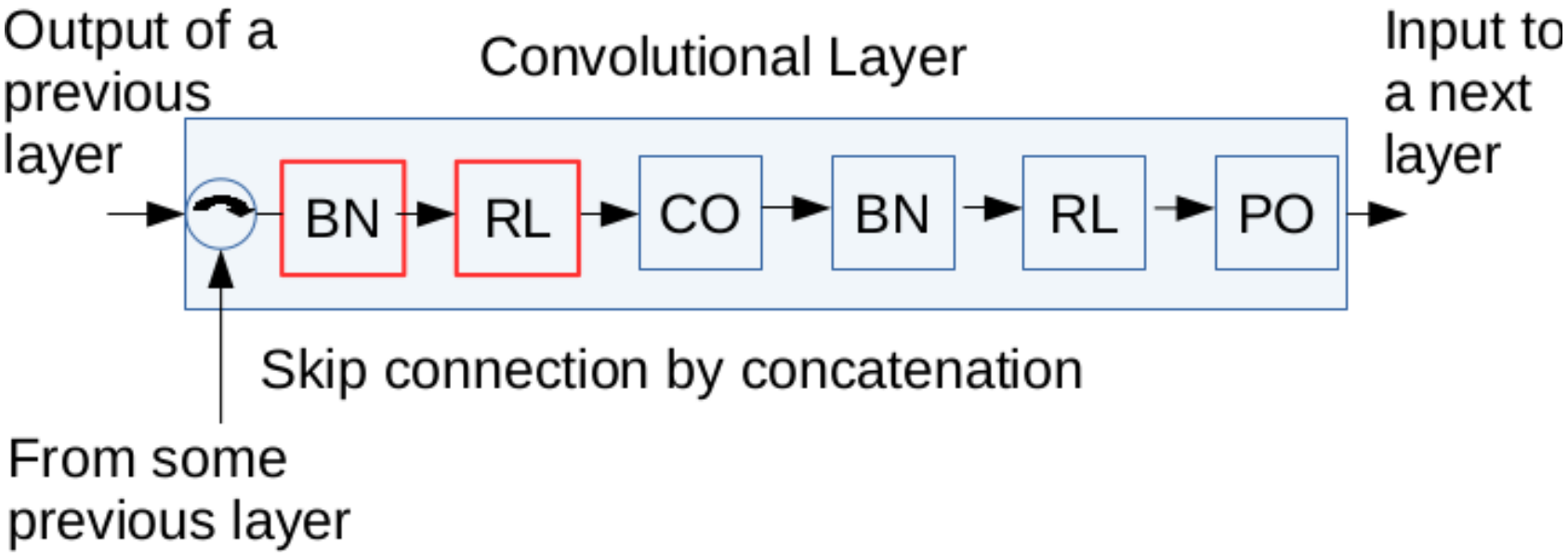
The filters are usually optimized by back-propagation using a MLP classifier but, deeper is the network, skip connections are required to avoid vanishing gradients and to recover the accuracy on some specific classes from previous layers.



Skip connections based on addition (e.g., ResNet) and concatenation (e.g., Dense Net) to the output of a previous layer.

Global feature space transformations

The filters are usually optimized by back-propagation using a MLP classifier but, deeper is the network, skip connections are required to avoid vanishing gradients and to recover the accuracy on some specific classes from previous layers.



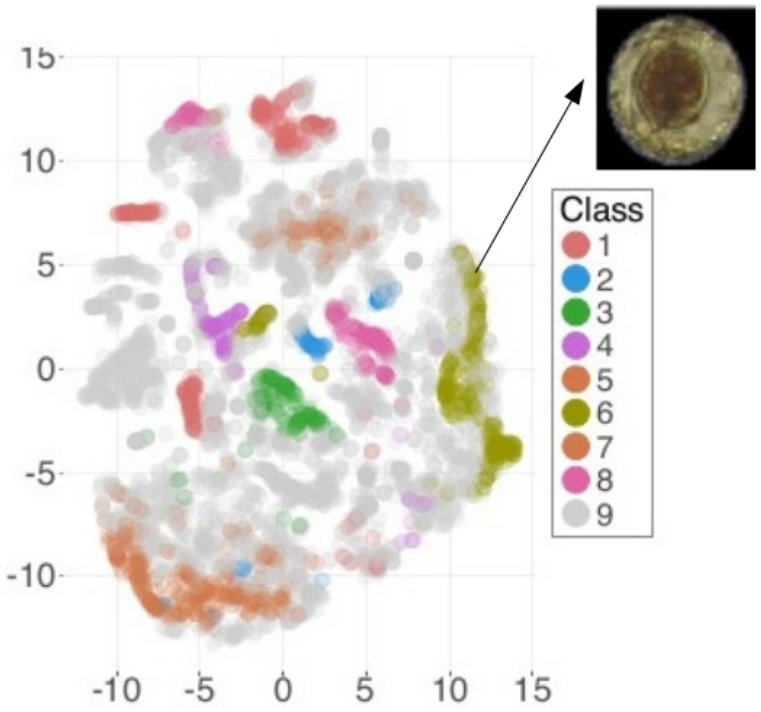
Skip connections based on addition (e.g., ResNet) and concatenation (e.g., Dense Net) to the output of a previous layer.

Agenda

- What are CNNs?
- Basic definitions and image processing operations.
- CNNs for image classification — how do they work?
- How to design, test, and visualize neuronal activity of a CNN in pytorch.
- Hands-on — image classification in pytorch.
- Final remarks.

CNNs for image classification

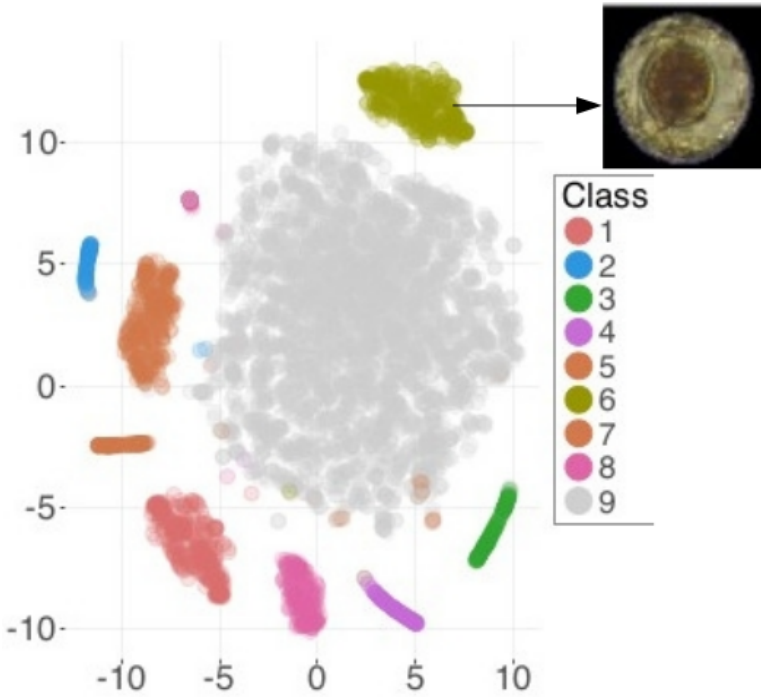
Starting from a global feature space in which the regions that separate the classes are not clearly defined.



Feature projection (t-SNE) before training a CNN for eight species of helminth eggs and impurities.

CNNs for image classification

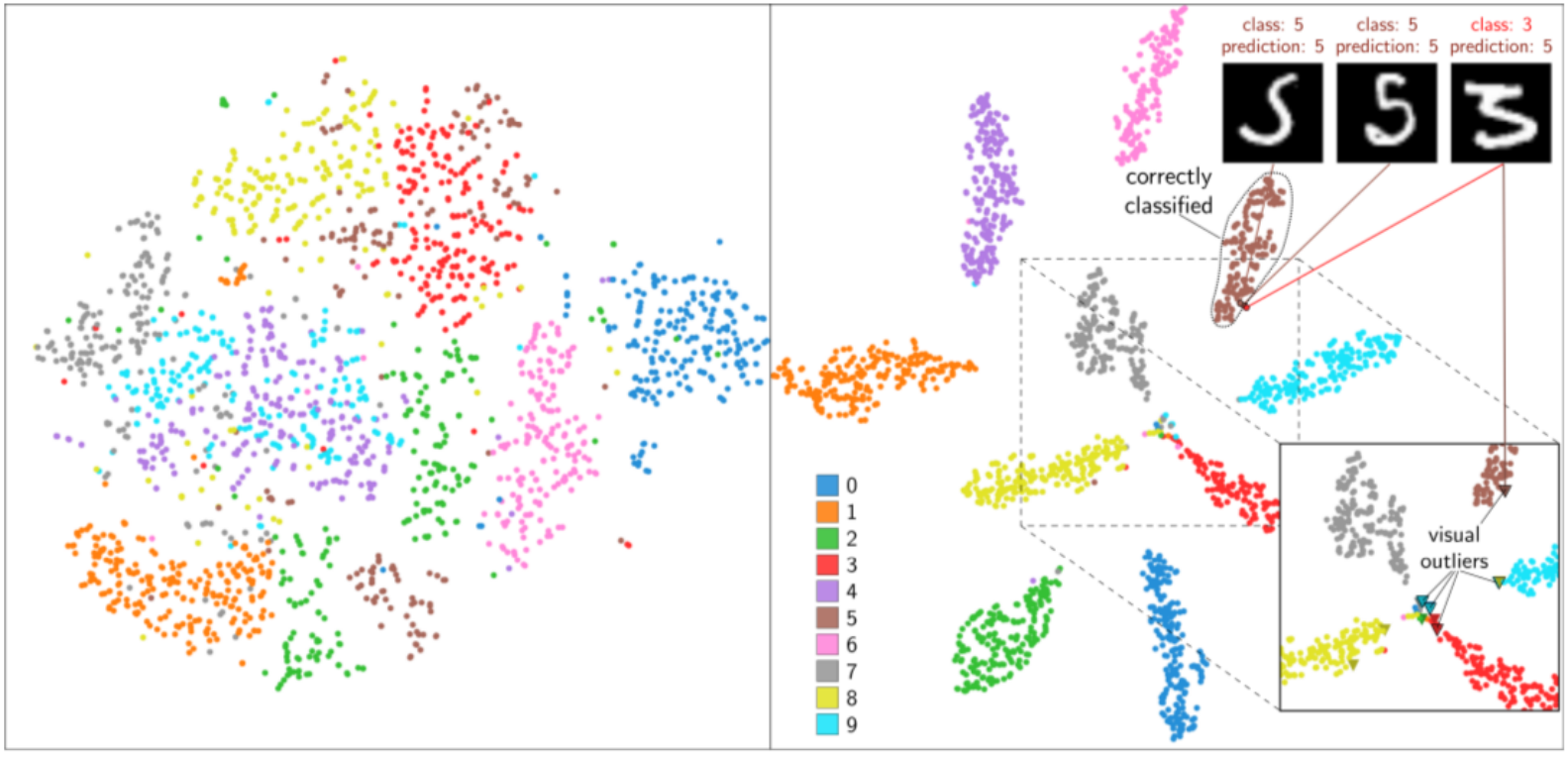
A clear definition of those regions is expected at the last hidden layer of the CNN.



Feature projection (t-SNE) after training a CNN for eight species of helminth eggs and impurities.

CNNs for image classification

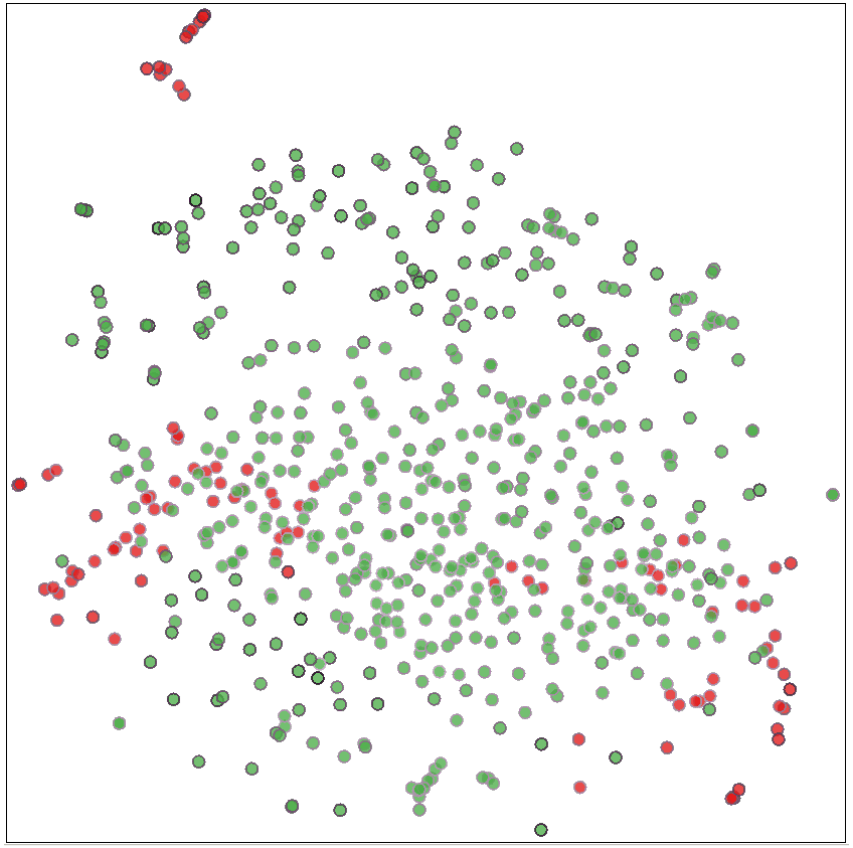
Another example with a known dataset of 10 digits, MNIST, in which one can understand the misclassification of outliers.



Feature projections (t-SNE) of the last hidden layer before (left) and after (right) training.

CNNs for image classification

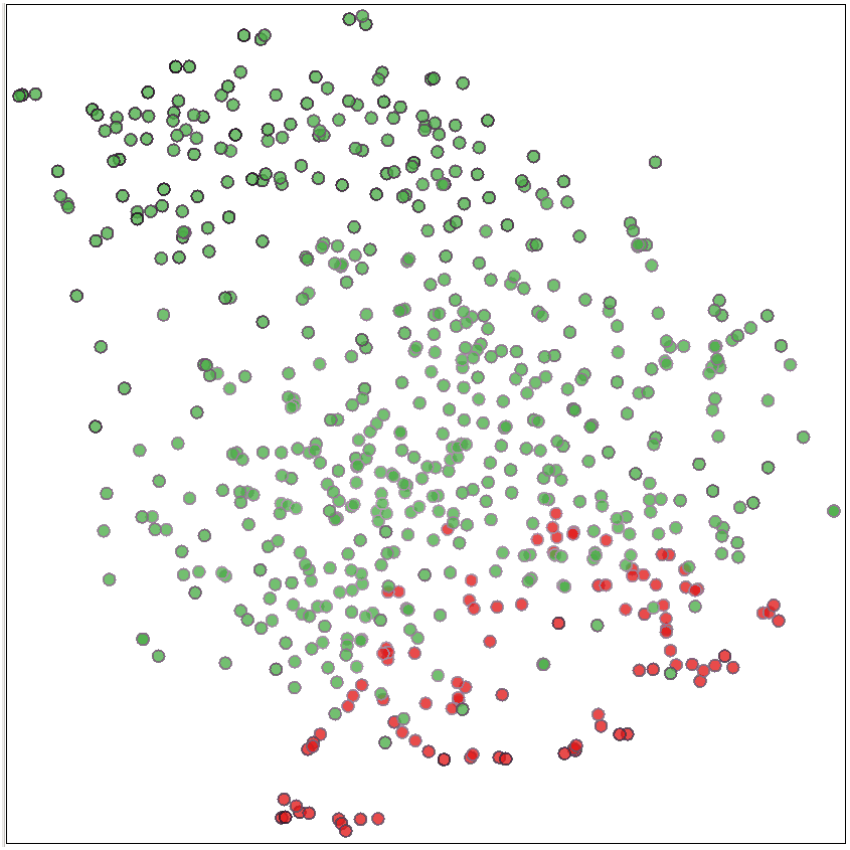
Even the global feature spaces at the outputs of deeper and subsequent convolutional layers should show progressively higher class separation.



Feature projections (t-SNE) after layers 10, 11, 12, and 13 for larvae of helminth and impurities.

CNNs for image classification

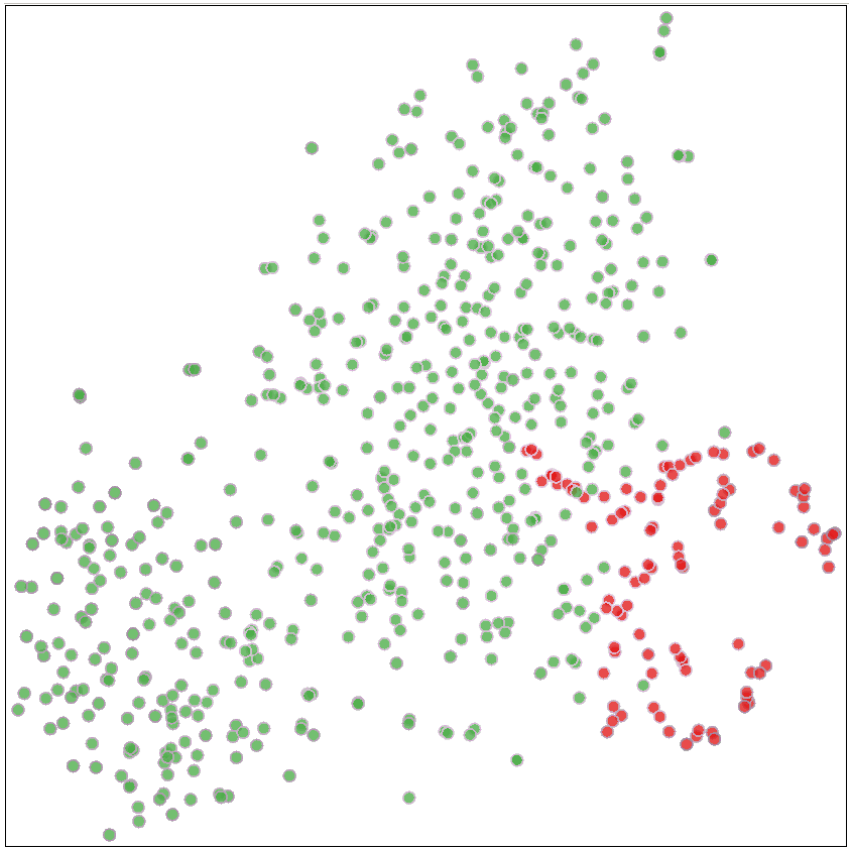
Even the global feature spaces at the outputs of deeper and subsequent convolutional layers should show progressively higher class separation.



Feature projections (t-SNE) after layers 10, 11, 12, and 13 for larvae of helminth and impurities.

CNNs for image classification

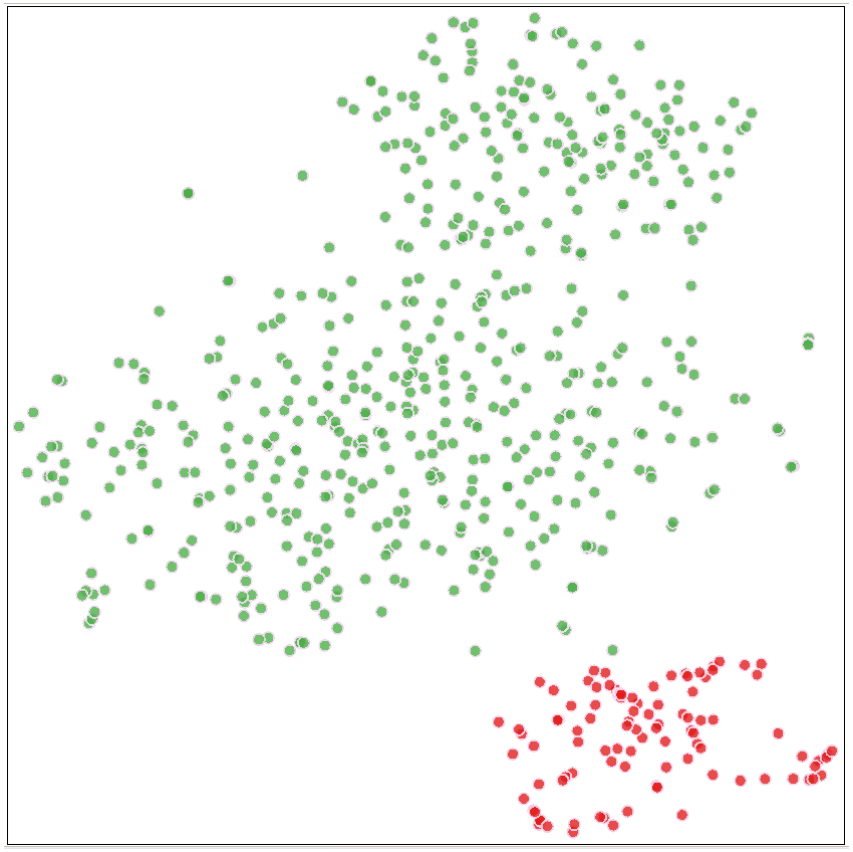
Even the global feature spaces at the outputs of deeper and subsequent convolutional layers should show progressively higher class separation.



Feature projections (t-SNE) after layers 10, 11, 12, and 13 for larvae of helminth and impurities.

CNNs for image classification

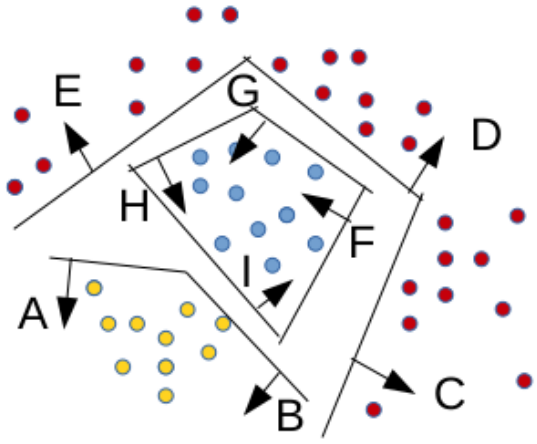
Even the global feature spaces at the outputs of deeper and subsequent convolutional layers should show progressively higher class separation.

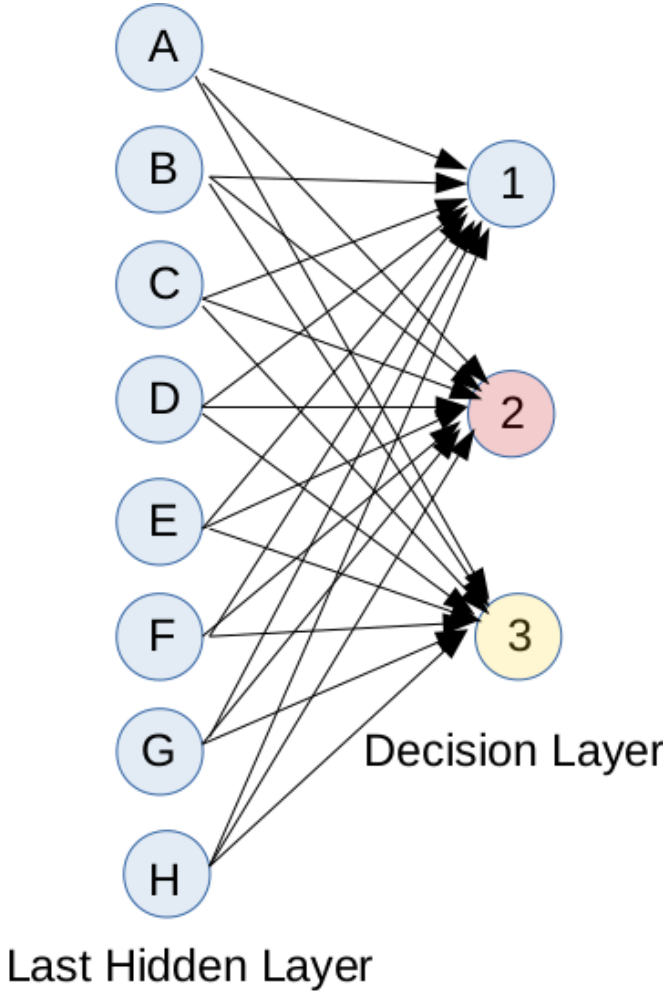


Feature projections (t-SNE) after layers 10, 11, 12, and 13 for larvae of helminth and impurities.

Class separation at the last hidden layer

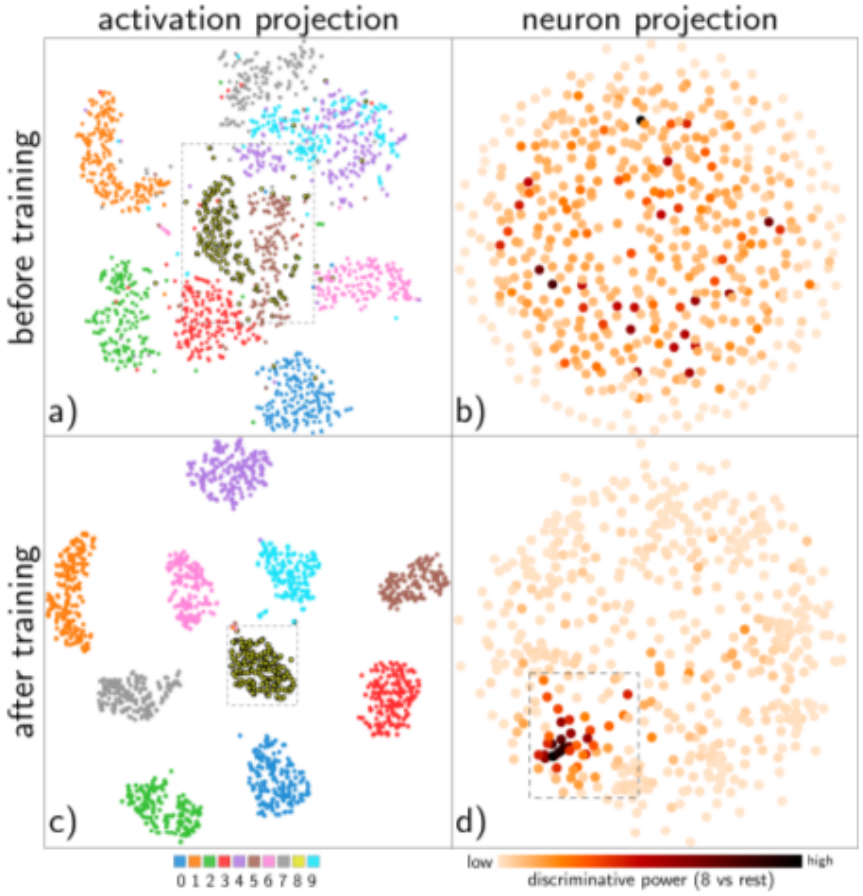
CNNs should then map images from distinct classes into different hyper-polyhedrons whose faces are the hyperplanes P of the last hidden layer's perceptrons.





Class separation at the last hidden layer

The training of a CNN hence specializes perceptrons of the last hidden layer in each given class.



Neuron projections (MDS, right) colored by their discriminative power for class 8 versus the others.

Agenda

- What are CNNs?
- Basic definitions and image processing operations.
- CNNs for image classification — how do they work?
- How to design, test, and visualize neuronal activity of a CNN in pytorch. (in python notebook by Italos Estilon de Souza)
- Hands-on — image classification in pytorch.
- Final remarks.

Final Remarks

- Research in deep learning is moving from dealing with training problems based on backpropagation (e.g., regularization tricks) to the design of explainable neural networks.
- In this context, visual analytics plays a crucial role in human-machine interaction during training, but feature projection methods need to improve in processing time and quality (see UMAP).
- It is possible to explore such methods in many tasks, such as data annotation, filter selection, neuronal layer assessment, etc.
- The ultimate goals for experts should be a better understanding of deep learning and the ability to intervene and improve the training process.

Selected works

1. G. Chiachia et al., Learning Person-Specific Representations From Faces in the Wild, doi: 10.1109/TIFS.2014.2359543 , IEEE TIFS, 9(12), 2089-2099, 2014.
2. D. Menotti et al., Deep Representations for Iris, Face, and Fingerprint Spoofing Detection, doi: 10.1109/TIFS.2015.2398817, IEEE TIFS, 10(4), 864-879, 2015.
3. P. E. Rauber et al., Visualizing the Hidden Activity of Artificial Neural Networks, doi: 10.1109/TVCG.2016.2598838, IEEE TVCG, 23 (1), 101-110, Jan. 2017.
4. P.E. Rauber et al., Visualizing Time-Dependent Data Using Dynamic t-SNE, `https://scholar.google.com/scholar?oi=bibs&cluster=18282991313013439645&btnI=1&hl=en`, EuroVis, 2016.

Selected works

1. P.E. Rauber et al., Projections as visual aids for classification system design. doi: 10.1177/1473871617713337, Information Visualization, 17(4), 282-305.
2. A. Z. Peixinho et al., Delaunay Triangulation Data Augmentation Guided by Visual Analytics for Deep Learning, doi: 10.1109/SIBGRAPI.2018.00056, SIBGRAPI, 384-391, 2018.
3. B. C. Benato, et al., Semi-Supervised Learning with Interactive Label Propagation Guided by Feature Space Projections, doi: 10.1109/SIBGRAPI.2018.00057, SIBGRAPI, 392-399, 2018.
4. R. Garcia et al., A Methodology for Neural Network Architectural Tuning Using Activation Occurrence Maps, doi: 10.1109/IJCNN.2019.8852223, IJCNN, 1-10, 2019.