Open for Innovation
KNIME

A Friendly Introduction to Codeless Deep Learning with KNIME Analytics Platform

Going live at:

Berlin 19:00 PM (UTC +2)

San Francisco 10:00 AM (UTC -7)

June 8, 2022

Meet the Speaker

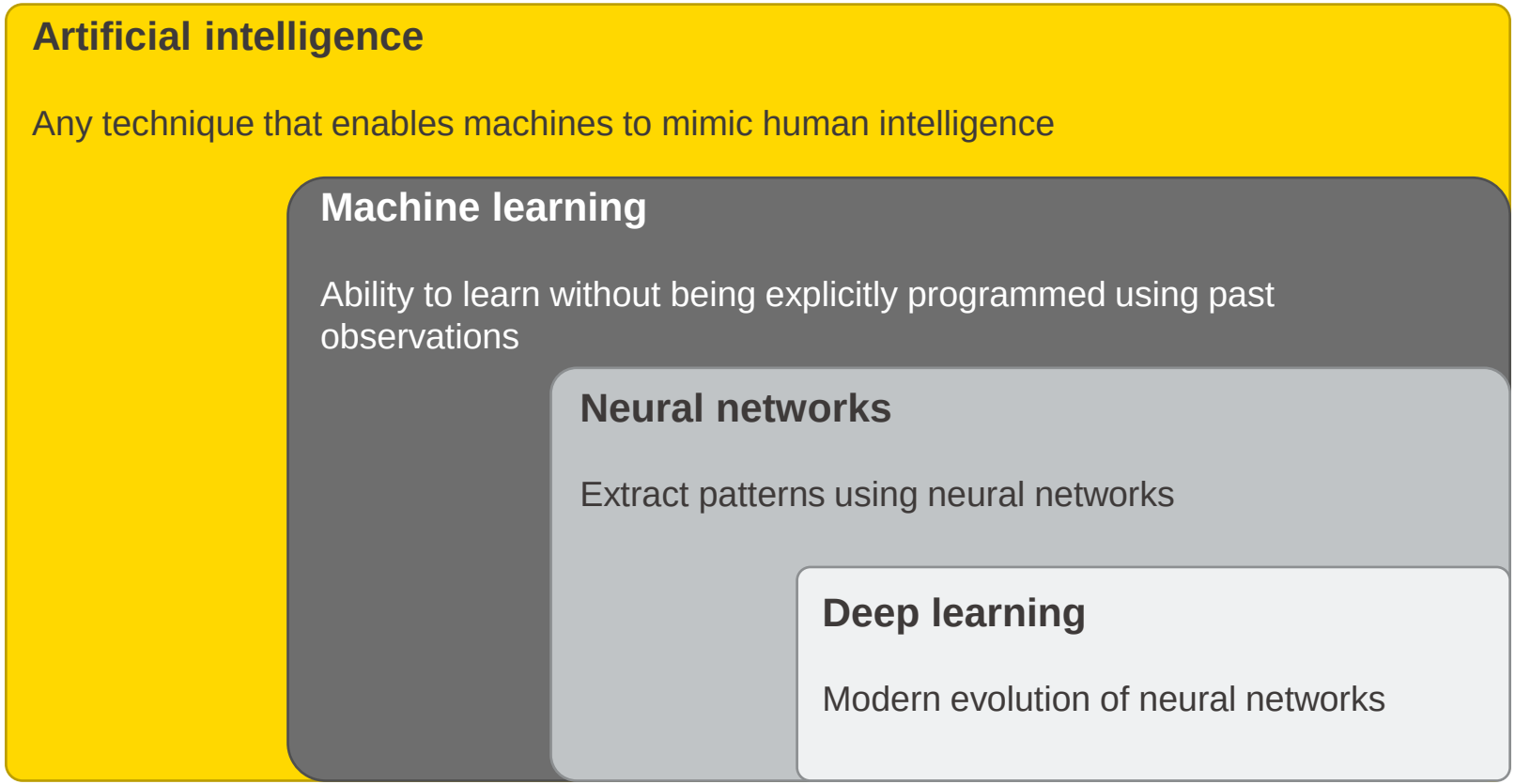
Roberto Cadili is a Data scientist on the Evangelism team at KNIME with a strong interest in machine learning algorithms and deep learning architectures for NLP and Computer Vision. As editor for Low Code for Advanced Data Science, he is helping the KNIME community shape successful data science stories, tutorials, and best practices that are worth sharing. He holds a Master's degree in Data Science and a Bachelor's degree in Economics.

LinkedIn – Roberto Cadili

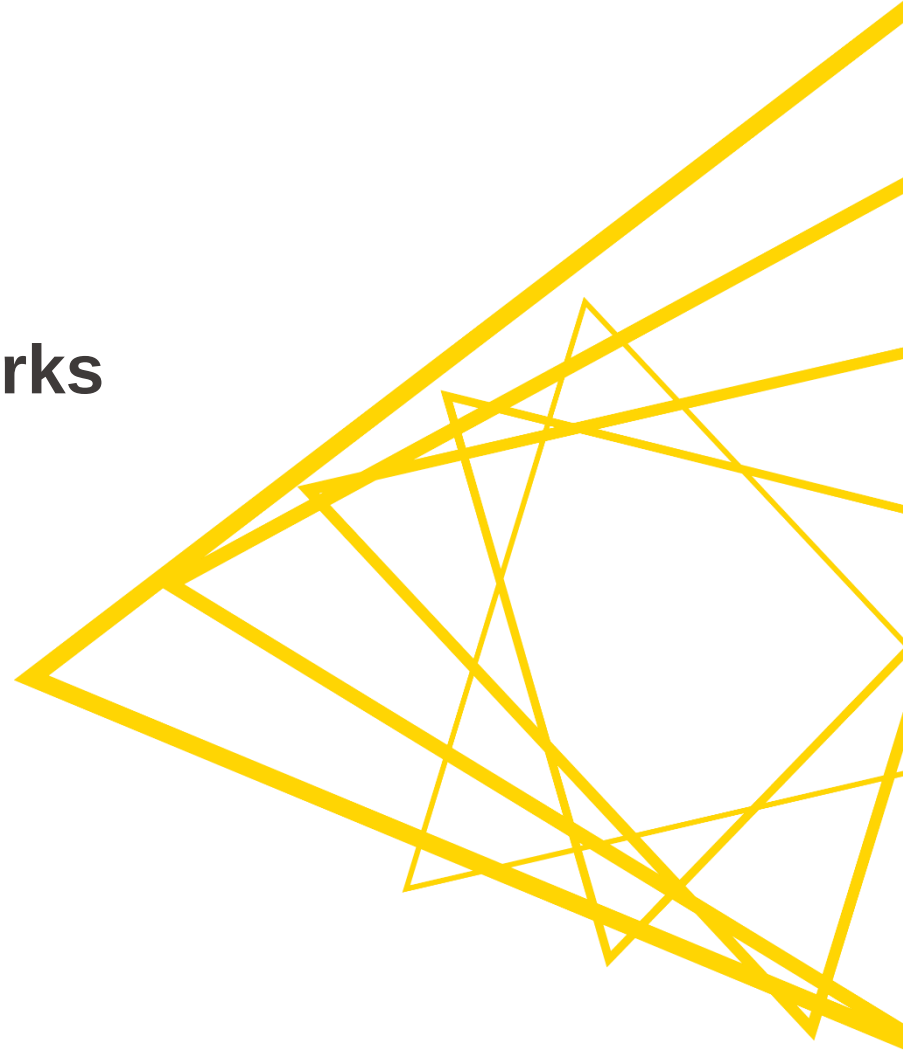
Twitter – @CadiliD

Email – roberto.cadili@knime.com

What is deep learning?

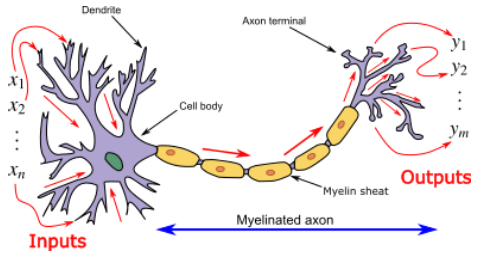


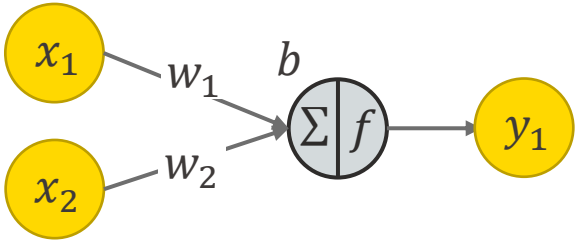
Basics Behind Neural Networks and Deep Learning



Let's Start Simple

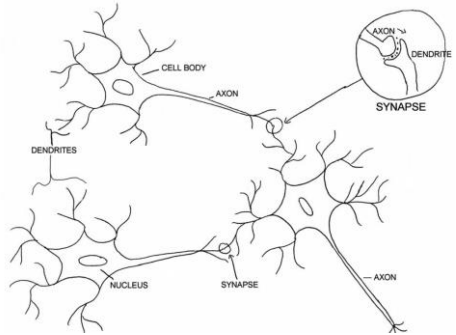
Single neuron

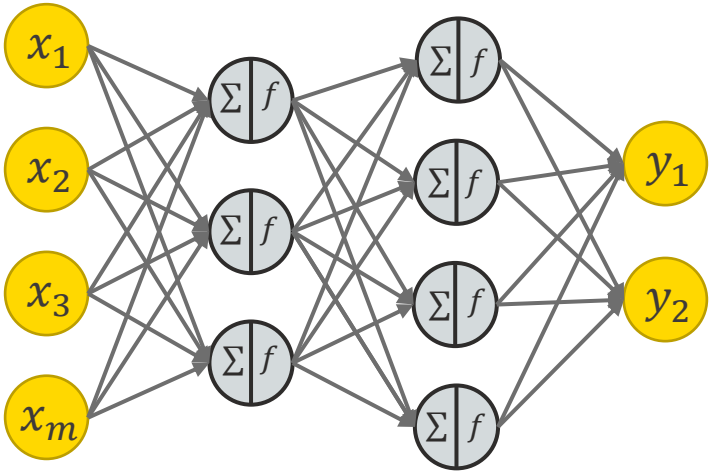




$$x_1 w_1 + x_2 w_2 + b$$

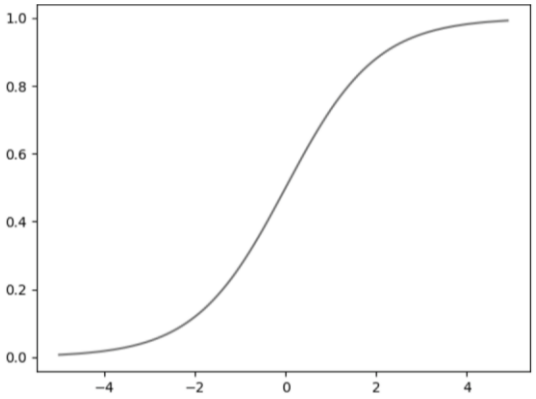
Neural network





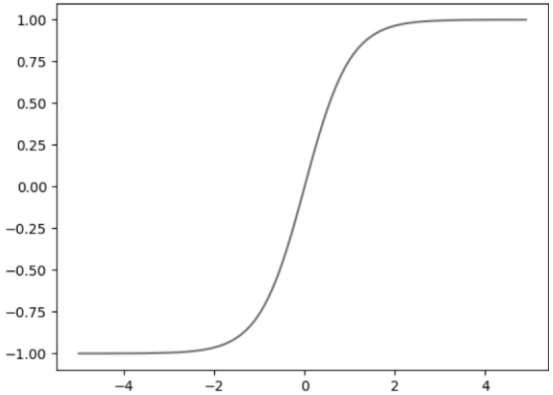
Frequently used Activation Functions

Sigmoid



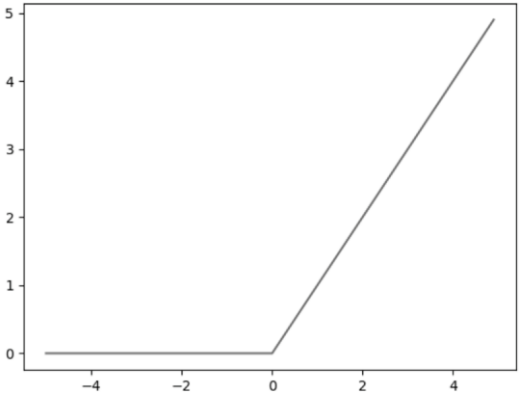
$$f(a) = \frac{1}{1 + e^{-a}}$$

Tanh



$$f(a) = \frac{e^{2a} - 1}{e^{2a} + 1}$$

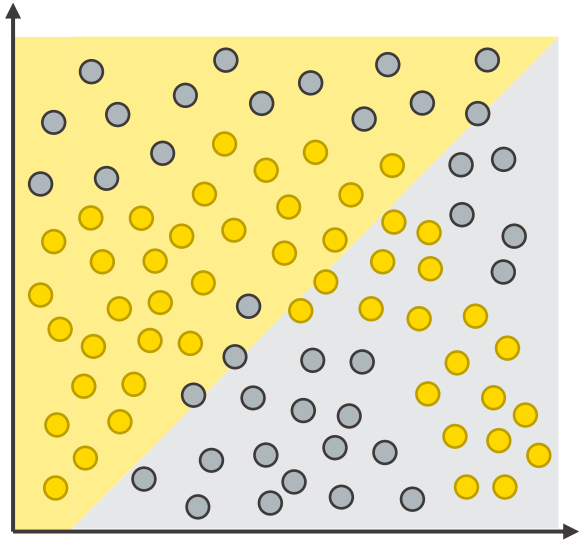
Rectified Linear Unit (ReLU)



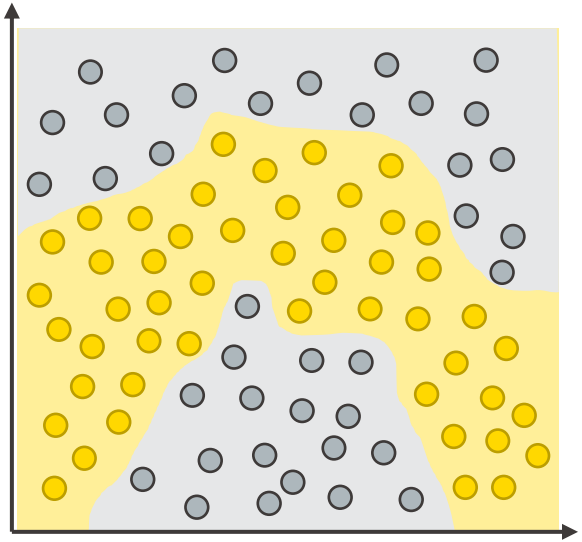
$$f(a) = max\{0, a\}$$

Why do we need non-linear activation functions?

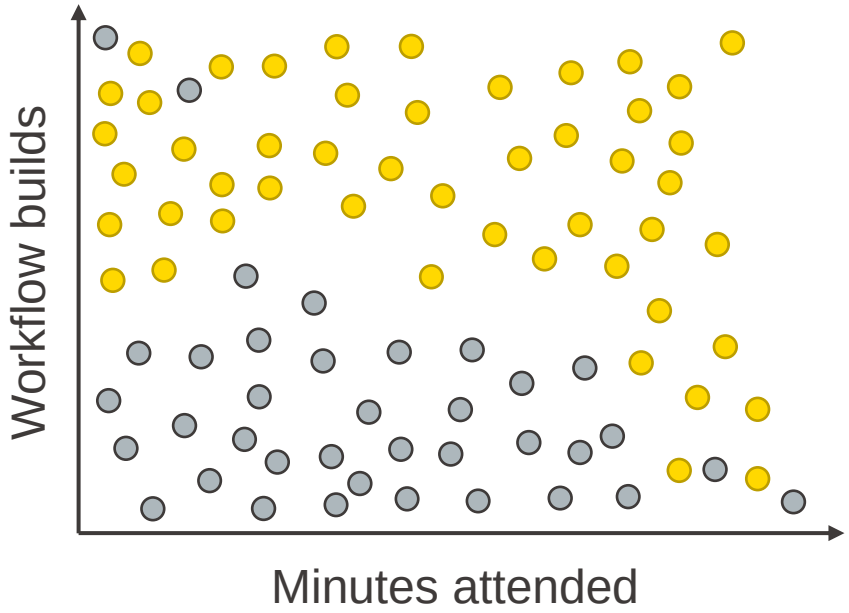
Linear



Non-linear

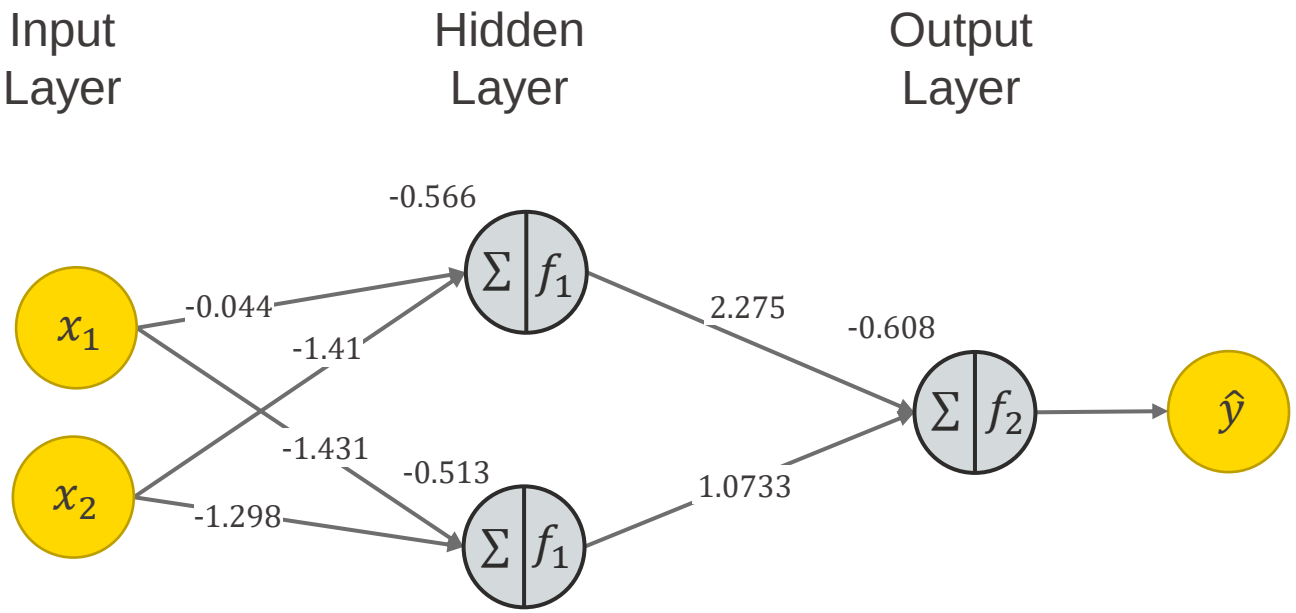


Example: Passing the KNIME L1-Certification



- Passed certification
- Didn’t pass certification

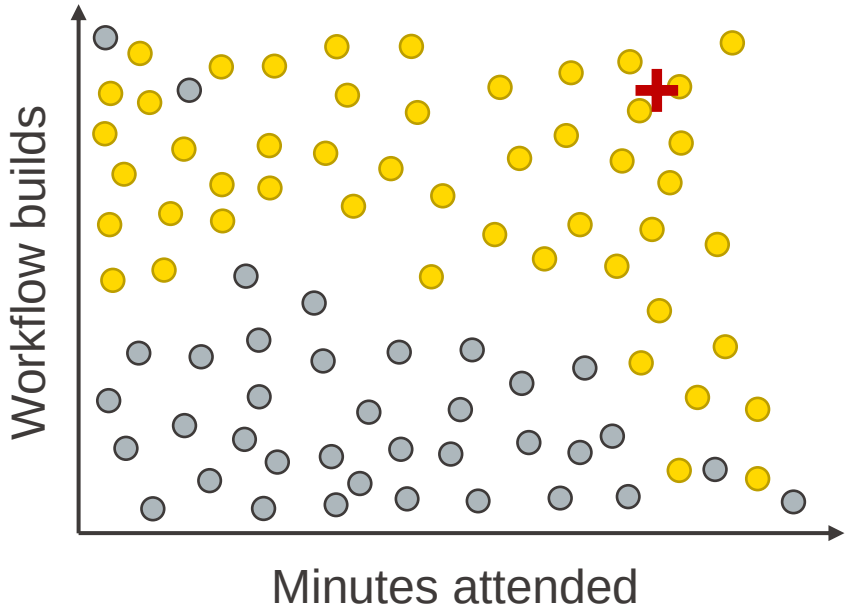
Example: Passing the KNIME L1-Certification



Input features:
x_1= minutes attended
x_2= workflows build

Output:
$\hat{y}$ =Probability that a person passed
$\hat{y} \geq 0.5 \Rightarrow Passed$ and $\hat{y} < 0.5 \Rightarrow Failed$

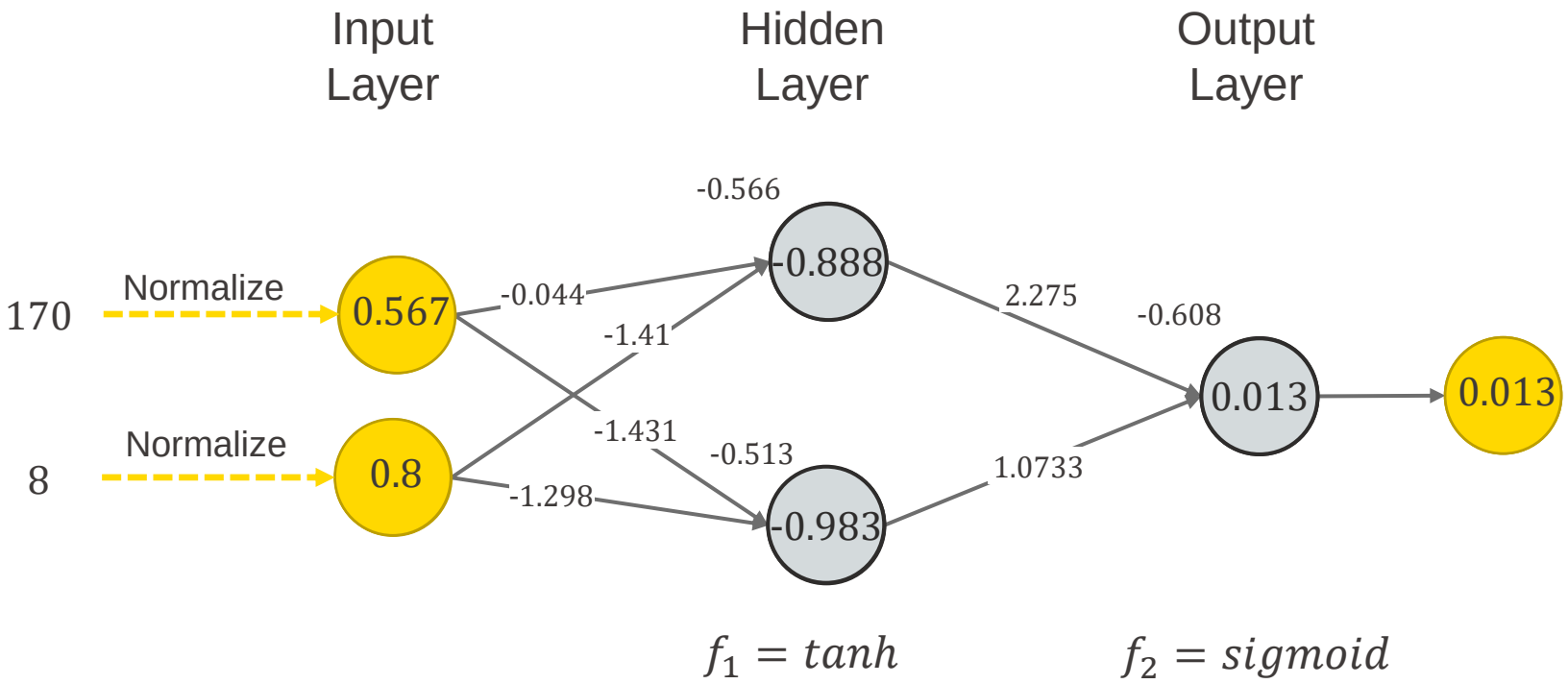
Example: Passing the KNIME L1-Certification



- Passed certification
- Didn’t pass certification
- New sample

x_1= minutes attended = 170
x_2= workflows build = 8

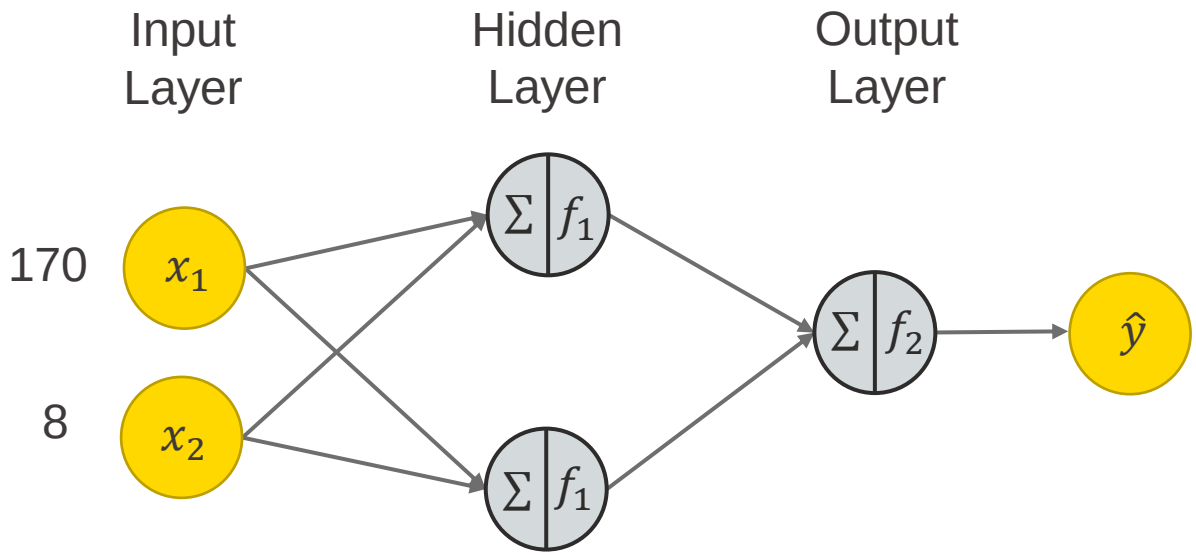
Example: Passing the KNIME L1-Certification



Input features:
x_1= minutes attended
x_2= workflows build

Output:
$\hat{y}$ =Probability that a person passed
$\hat{y} \geq 0.5 \Rightarrow Passed$ and $\hat{y} < 0.5 \Rightarrow Failed$

Training a Neural Network = Finding Good Weights



Predicted $\hat{y}$ = 0.013
True y = 1

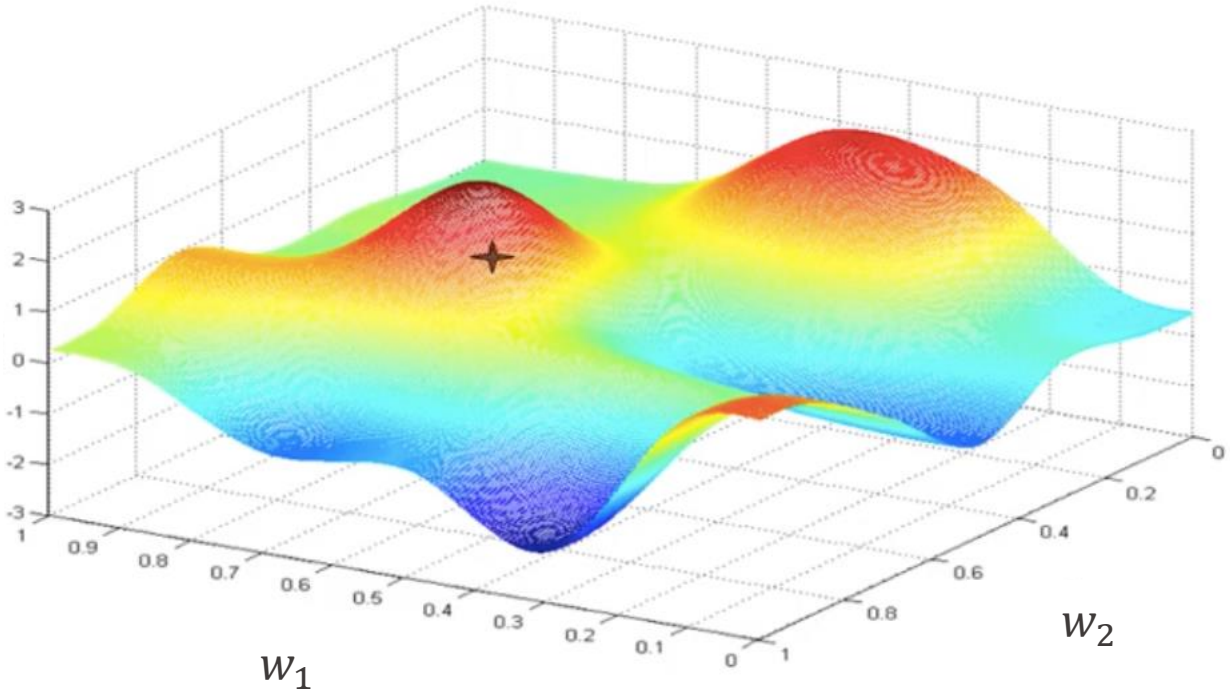
Binary cross entropy

$$\mathcal{L} = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

$$J(W) = \sum \mathcal{L}\left(\hat{y}(x_1, x_2, W), y\right)$$

Example of a Loss Landscape

$J(w_1, w_2)$



Goal find w_1 and w_2 of the global minima of the loss landscape

Idea Behind Gradient Descent

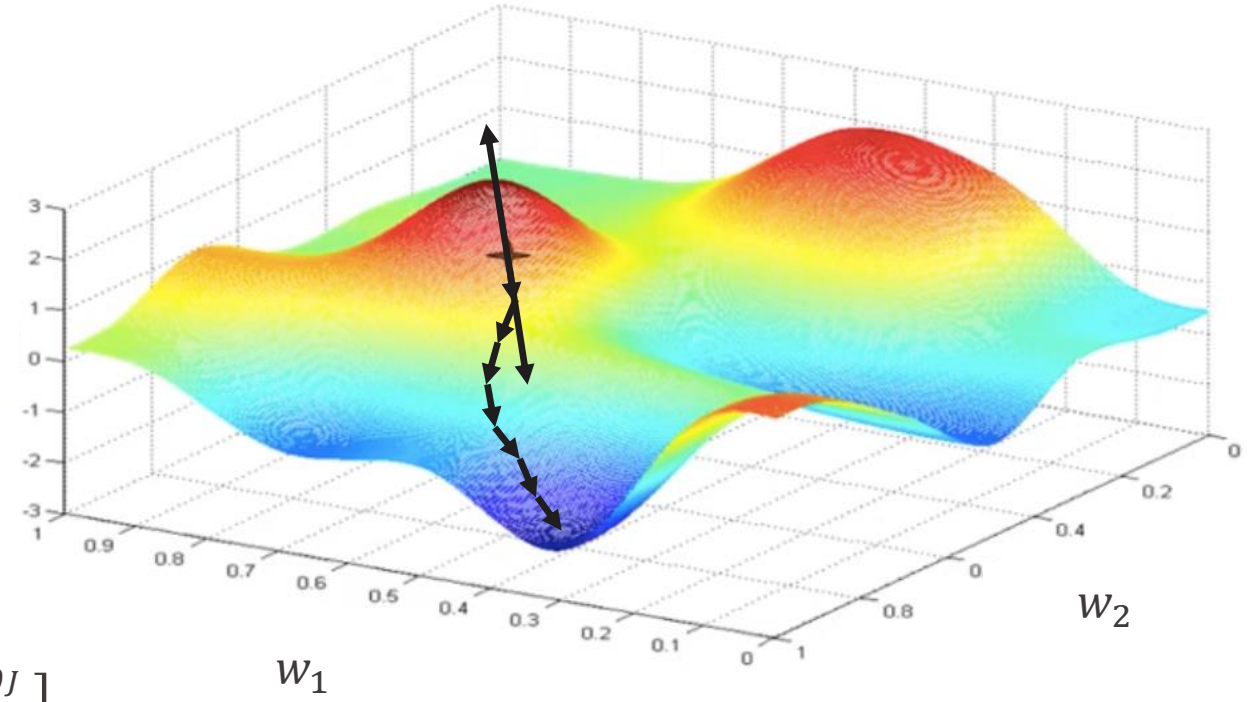
Mountain cycling in foggy conditions	Gradient descent algorithm

1. Start at your current position
2. Until you reached the valley
  1. Look for the direction of steepest ascent
  2. Cycle into the opposite direction for 2m
3. Return the position of the valley
	1. Initialize the weights W
	2. Until we reach a minimum
	  1. Calculate the gradient with respect to the weights $\nabla_W J(x, W)$
	  2. make a little step into the opposite direction $W \leftarrow W - \eta \nabla_W J(x, W)$
	3. Return the weights

Note: $\nabla_W J(x, W) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \end{bmatrix}$, vector with the partial derivatives with respect to all weights.

Idea Behind Gradient Descent

$J(w_1, w_2)$

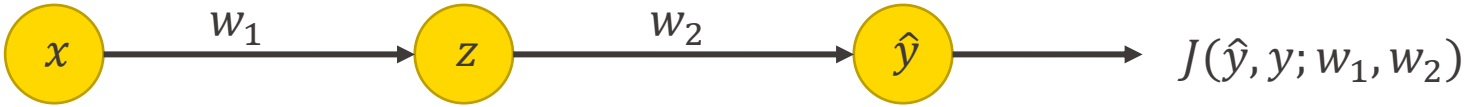


$$\nabla_W J(x, W) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \end{bmatrix}$$

Backpropagation

- Efficient way to calculate the gradient during optimization

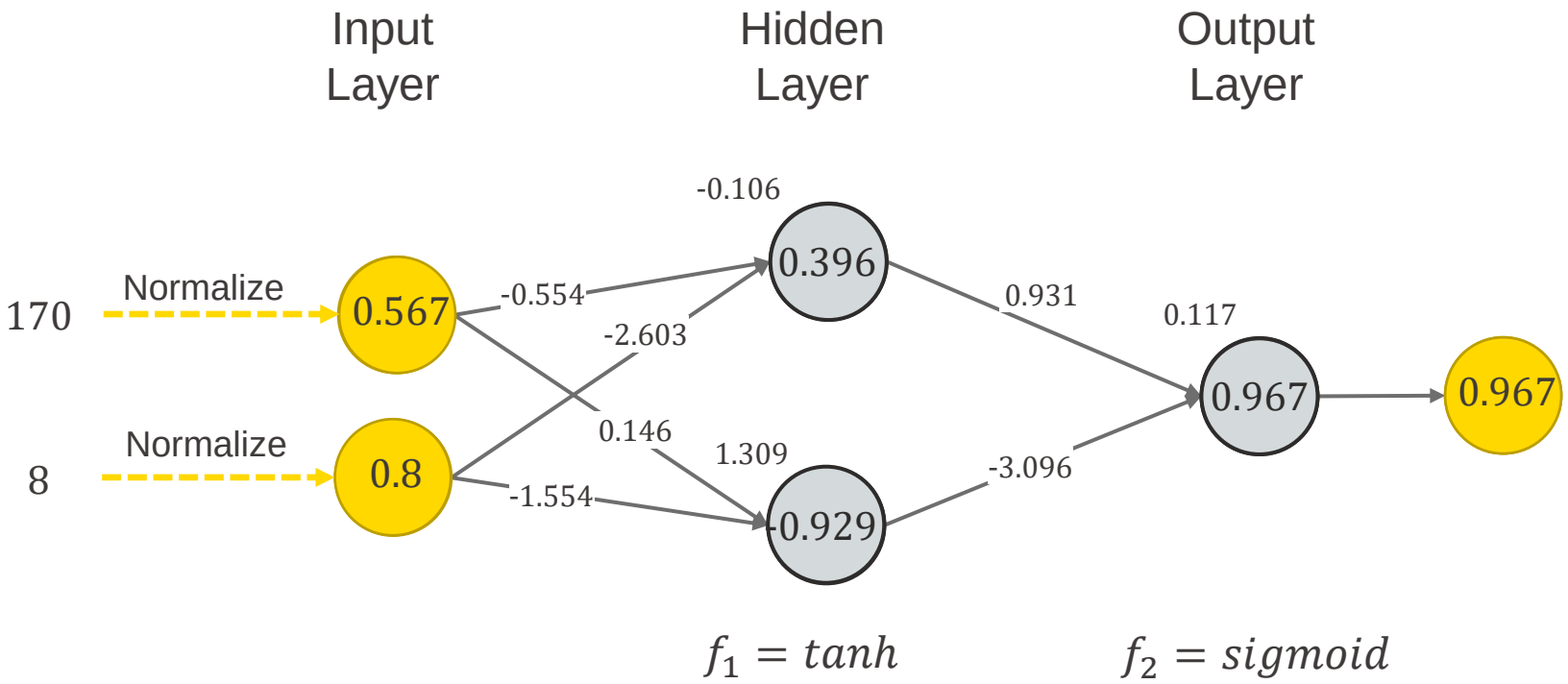
Forward pass



Backward pass

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w_2} \qquad \frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z} * \frac{\partial z}{\partial w_1}$$

Example: Passing the KNIME L1 Certification



Input features:
x_1= minutes attended
x_2= workflows build

Output:
y = Probability that a person passed
$y \geq 0.5 \Rightarrow Passed$ and $y < 0.5 \Rightarrow Failed$

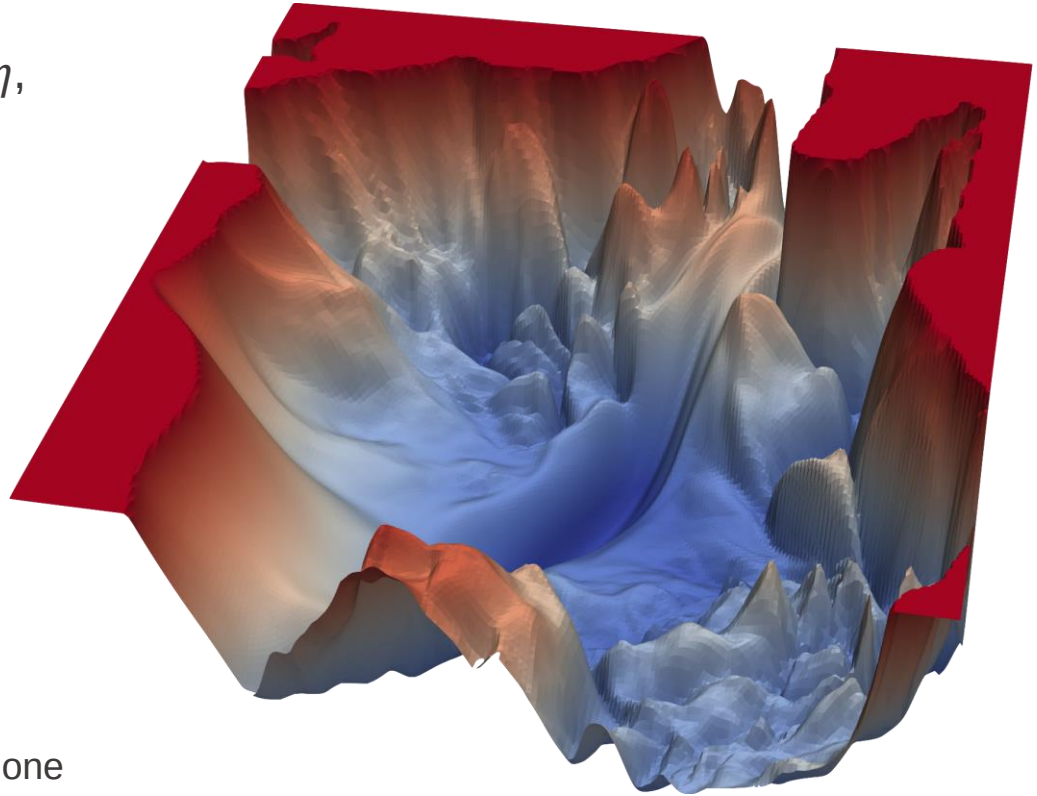
Loss Landscape of a Real Neural Network

- A good choice for the step size η, aka learning rate, is important

$$W \leftarrow W - \eta \nabla_W J(x, W)$$

- Many different optimizers with adaptive learning rates are available
  - Adam, Adadelta, Adagrad, ect

- Other important settings
  - Batch size, aka number of samples for one update
  - Number of epochs, aka how often each sample is used



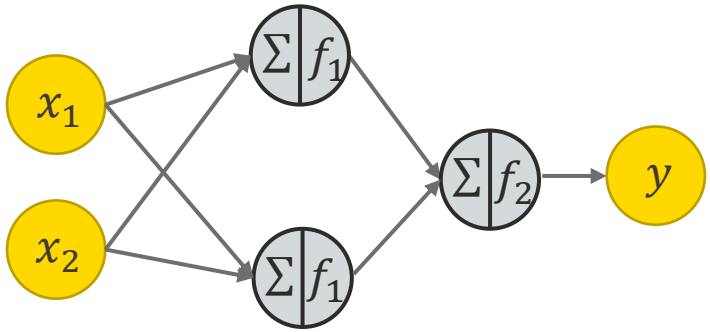
Source: https://www.cs.umd.edu/~tomg/projects/landscapes/

Different Loss Functions

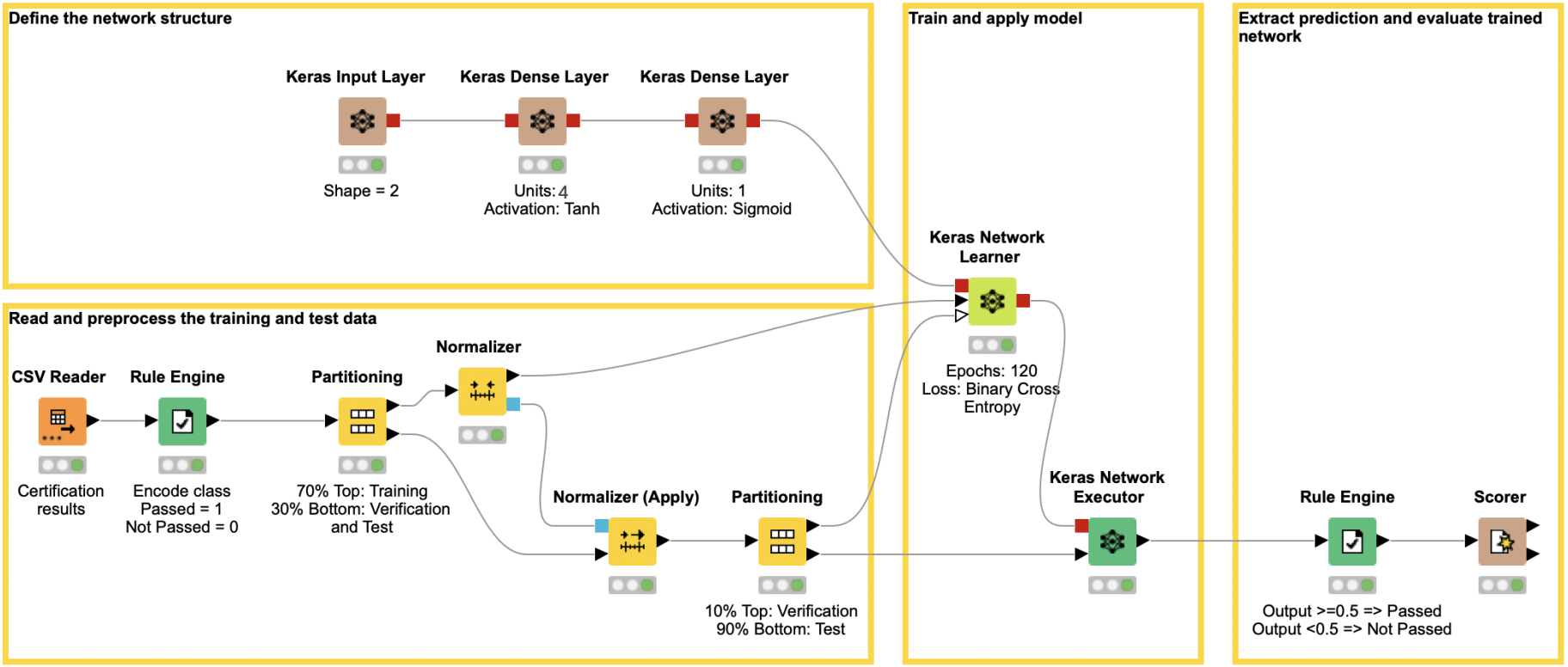
- Binary classification
  - Binary cross entropy
  - Tip 1: Encode the two classes a 0 and 1
  - Tip 2: Use sigmoid as activation in the last layer with 1 unit
- Multi-class classification
  - Categorical cross entropy
  - Tip: Use softmax as activation in the last layer with the number of units equal to the number of different classes
- Regression problem
  - Mean squared error
  - Mean absolute error
  - Tip: Use linear or ReLu as activation in the last layer

Settings for a Feed Forward Neural Network

- The network structure
  - How many layers
  - For each layer:
    - Number of neurons
    - Activation function
- Loss function
- Optimizer and settings

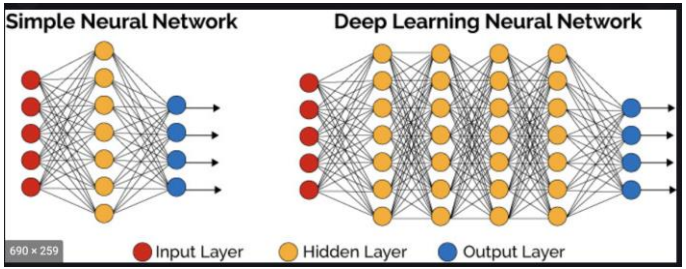


Codeless Deep Learning with KNIME Analytics Platform



From Neural Network To Deep Learning

- Deep feed forward network

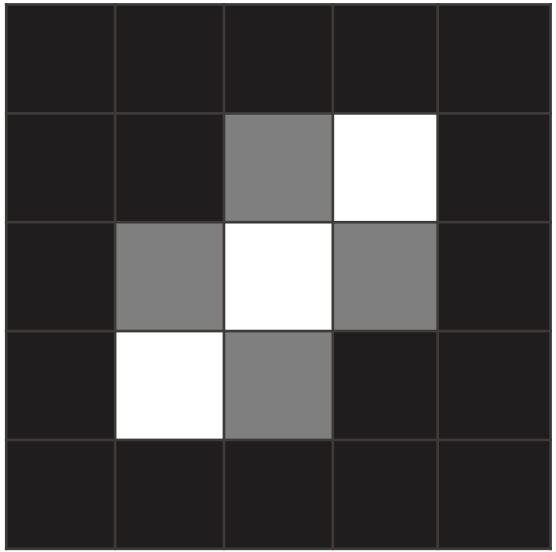


- Many additional layer types
  - Convolutional Layers for Images (e.g. Alex Net)
    - Image classification, Image segmentation
  - Recurrent Layers for sequential data
    - Time series prediction, language models, neural machine translation
    - Topic of the next webinar
- New architectures
  - GANs
  - Transformer networks

MNIST Image Classification

- Goal: Classify handwritten digits between 0 and 9

How Can We Represent a Gray-Scale Image?



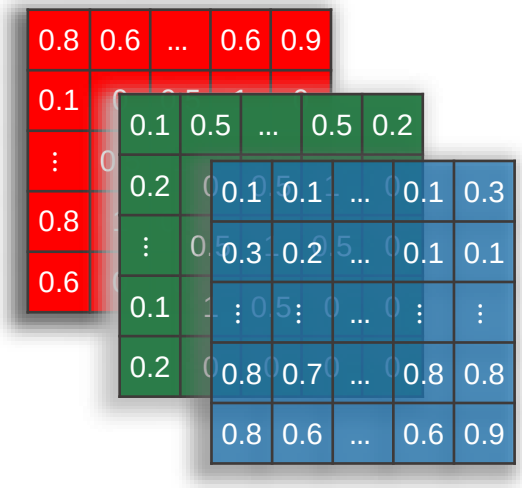
=

0	0	0	0	0
0	0	0.5	1	0
0	0.5	1	0.5	0
0	1	0.5	0	0
0	0	0	0	0

- A gray-scale image can be stored in a matrix
- Each cell represents one pixel of the image

How Can We Represent a Colored Image?

=



- A colored image can be encoded via the intensity of red, green, and blue for each pixel.
  => It can be stored in a tensor with one channel for each color
  - Example: n x m pixel image with k channels can be stored in a tensor of size n x m x k.

Use Filters to Check for Different Features

Check for arms going from lower right to top left

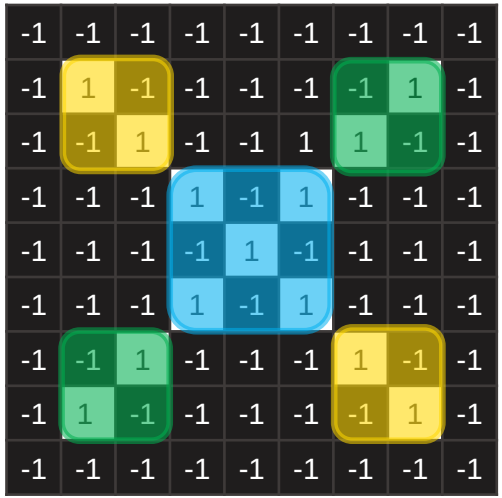
1	-1
-1	1

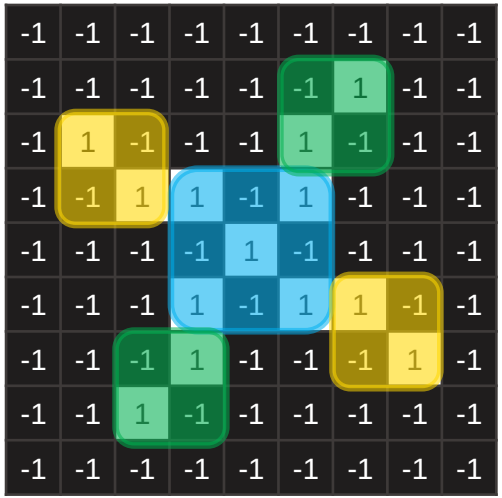
Check for crosses

1	-1	1
-1	1	-1
1	-1	1

Check for arms going from lower left to top right

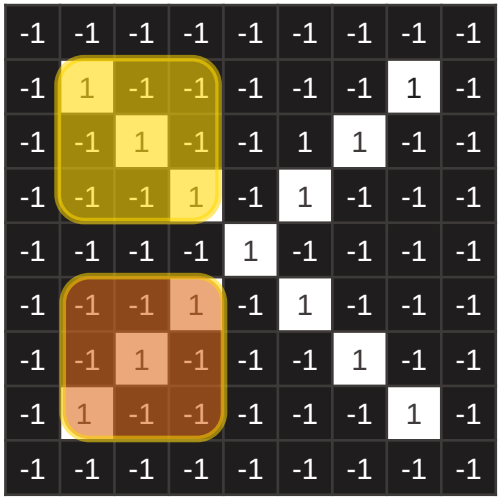
-1	1
1	-1





How Can We Apply a Filter

- Goal of a filter:
  - High value if the feature is in an image patch
  - Low value if the feature is not in an image patch
- Idea:
  - Use a kernel / matrix and place it on top of different parts of the image
  - Multiply the pixel value with the according kernel value and sum up the values



$*$

1	-1	-1
-1	1	-1
-1	-1	1

$=$

$$1 * 1 + (-1) * (-1) + (-1) * (-1) + (-1) * (-1) + 1 * 1 + (-1) * (-1) + (-1) * (-1) + (-1) * (-1) + 1 * 1 = 9$$

$*$

1	-1	-1
-1	1	-1
-1	-1	1

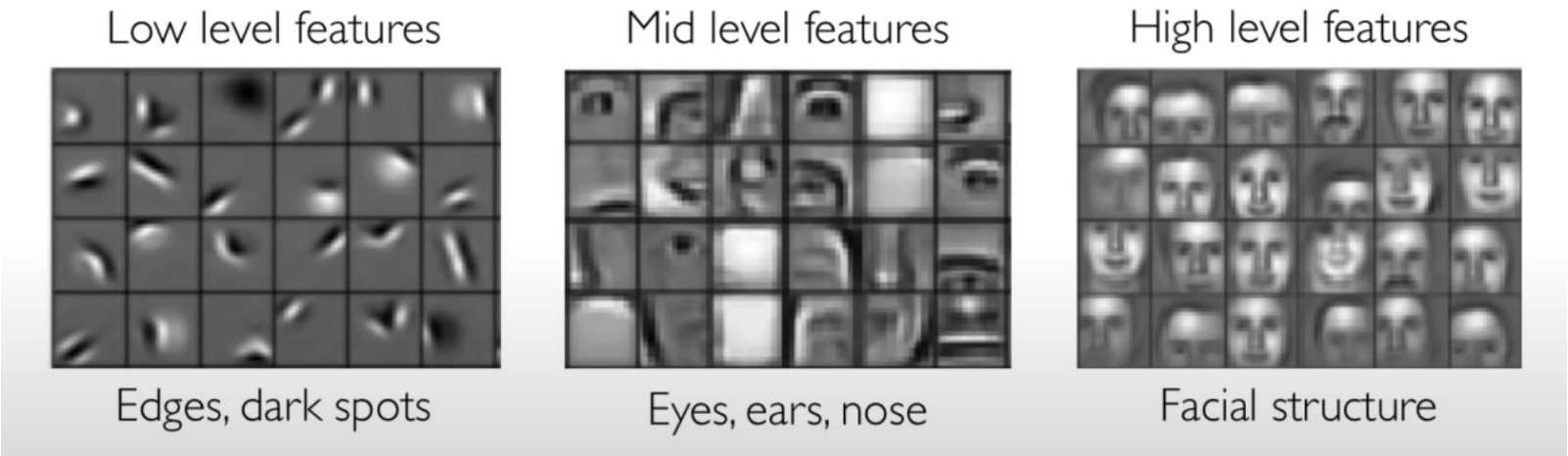
$=$

$$(-1) * 1 + (-1) * (-1) + 1 * (-1) + (-1) * (-1) + 1 * 1 + (-1) * (-1) + 1 * (-1) + (-1) * (-1) + (-1) * 1 = 1$$

Note: In the deep learning community this operation is called a convolution and is represented via an asterisk $*$. Strictly mathematical it is a cross correlation.

Convolutional Neural Network (CNN)

- A CNN is a neural network with at least one convolutional layer.
- Instead of handcrafting different features a CNN learns a hierarchy of features using multiple convolution layers that detect different features.



Images from: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L3.pdf

How Do Convolutional Layers Works?

- Idea: Instead of connecting every neuron to the new layer a sliding window is used.

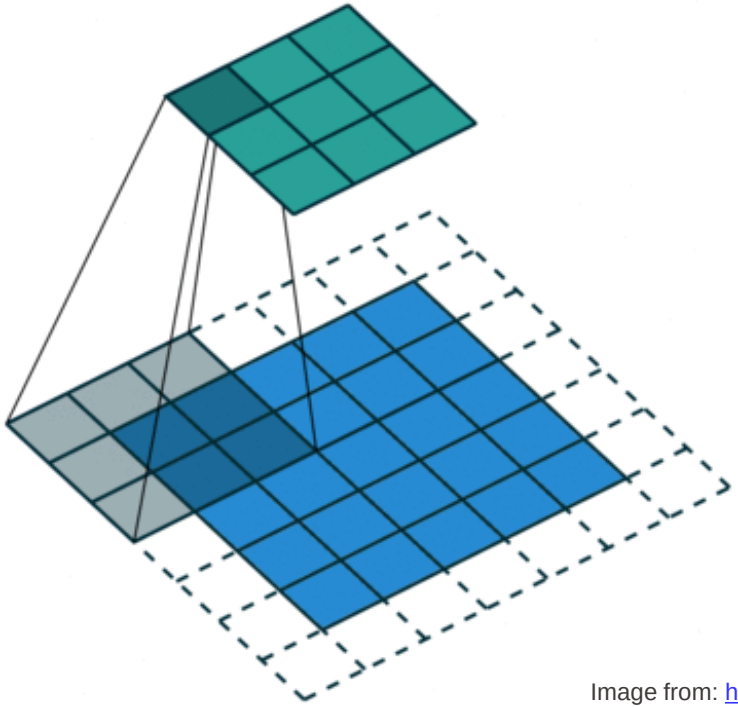
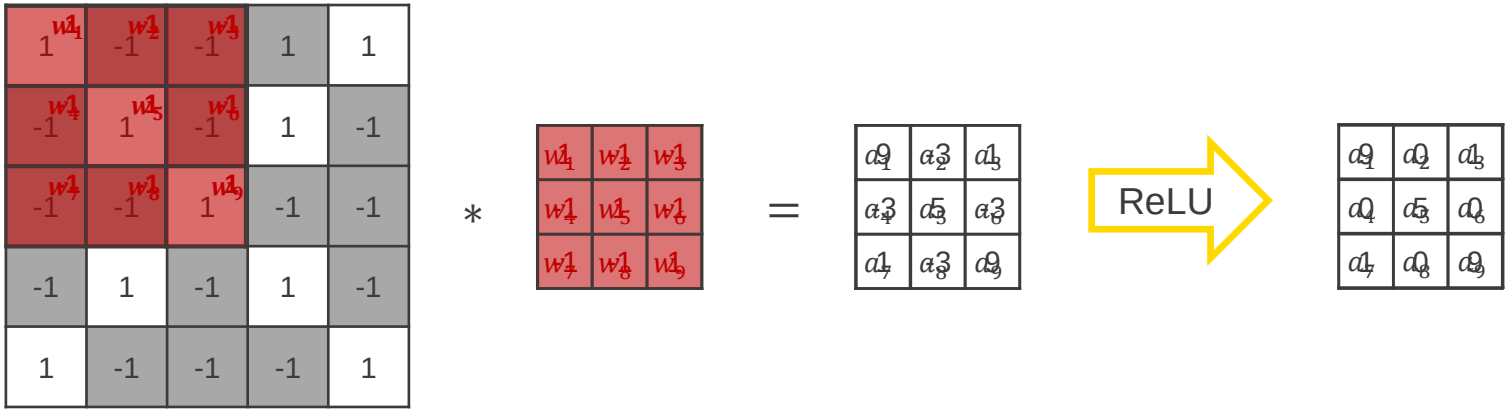


Image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

How Do Convolutional Layers Work?

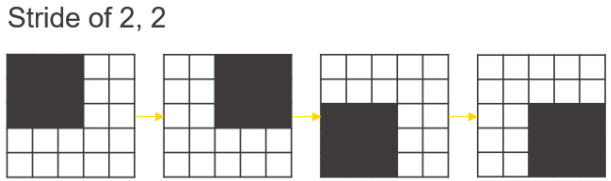
- Idea:
  - Use a kernel / weight matrix and slide it over the image
  - At each position: Apply the convolution and a non-linear activation, e.g. ReLU



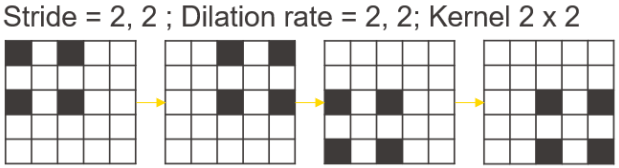
- The weights of the kernel are learned during training
- Note: These are similar steps like in a feed forward neural network
  - Convolution ≙ Weighted sum of inputs

Convolutional Neural Networks (CNN)

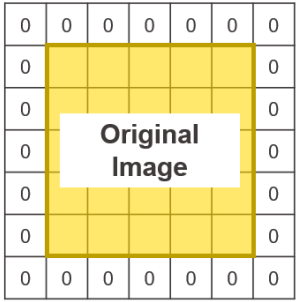
- Stride
  - The *jump* the kernel makes when moving over the input image
  - Reduces the spatial resolution



- Dilation rate
  - If kernels are not contiguous, the spaces between each cell are called dilation
  - Reduces the spatial resolution

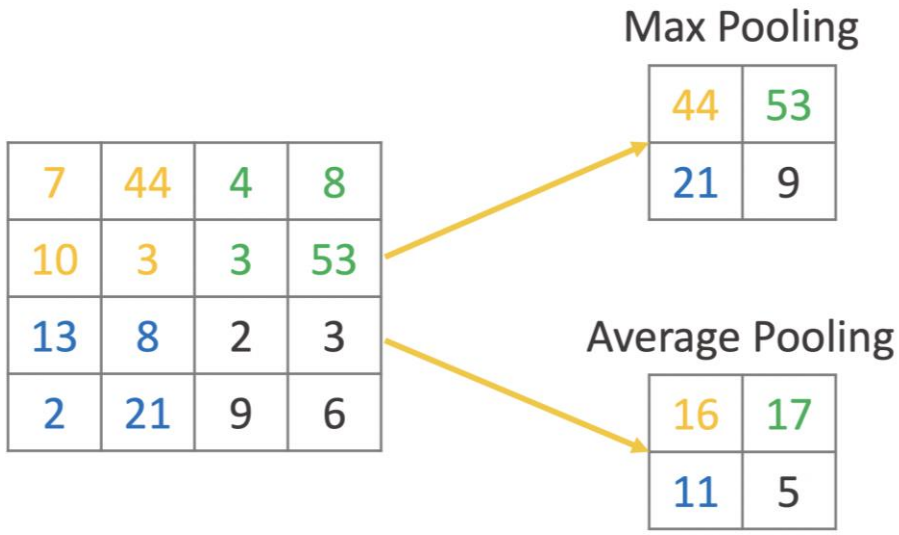


- Padding
  - Artificially increases the input at the boundary
  - Helps with preserving the spatial resolution and alignment



Pooling Layer

- Idea: Replace the area of an image or feature map with a summary statistic.
- Example: Replace each 2x2 area with the
  - Maximum value (Max pooling)
  - Mean value (Average pooling)
- Pooling layers are often used to reduce the spatial resolution in between convolutional layers to
  - Increase the receptive field of the following layers
  - Reduce computational complexity



Classification Layers

- After the sequence of convolutional + pooling layers, a classic fully connected feedforward neural network is applied to carry out the classification process.

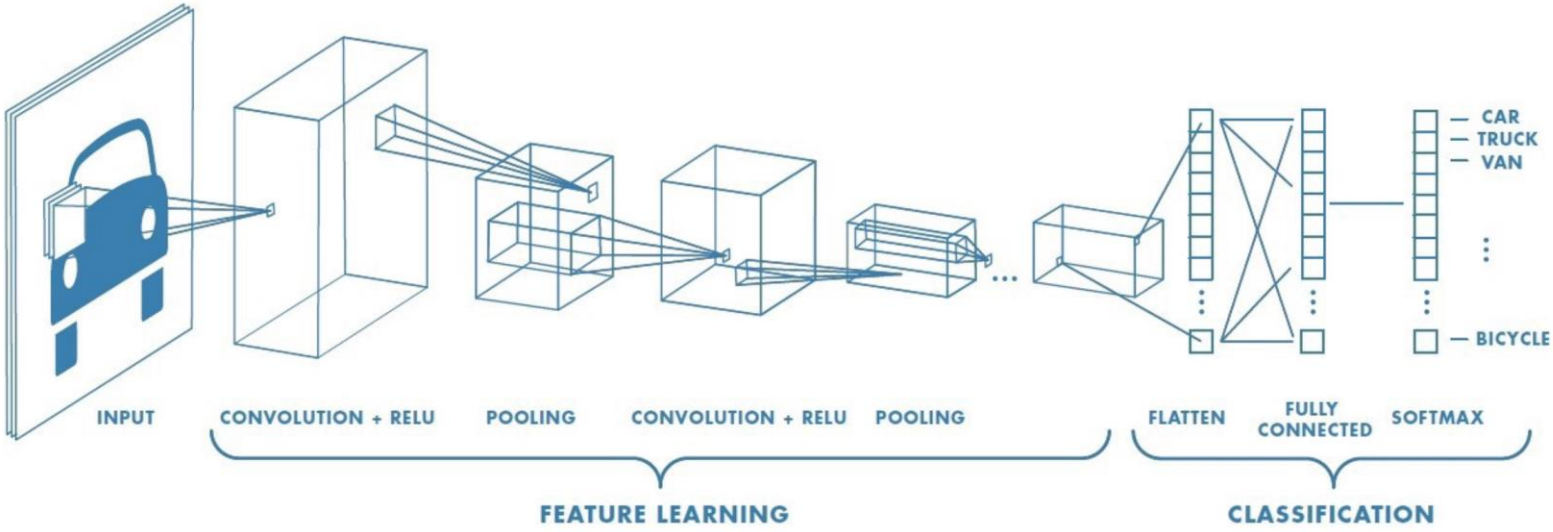
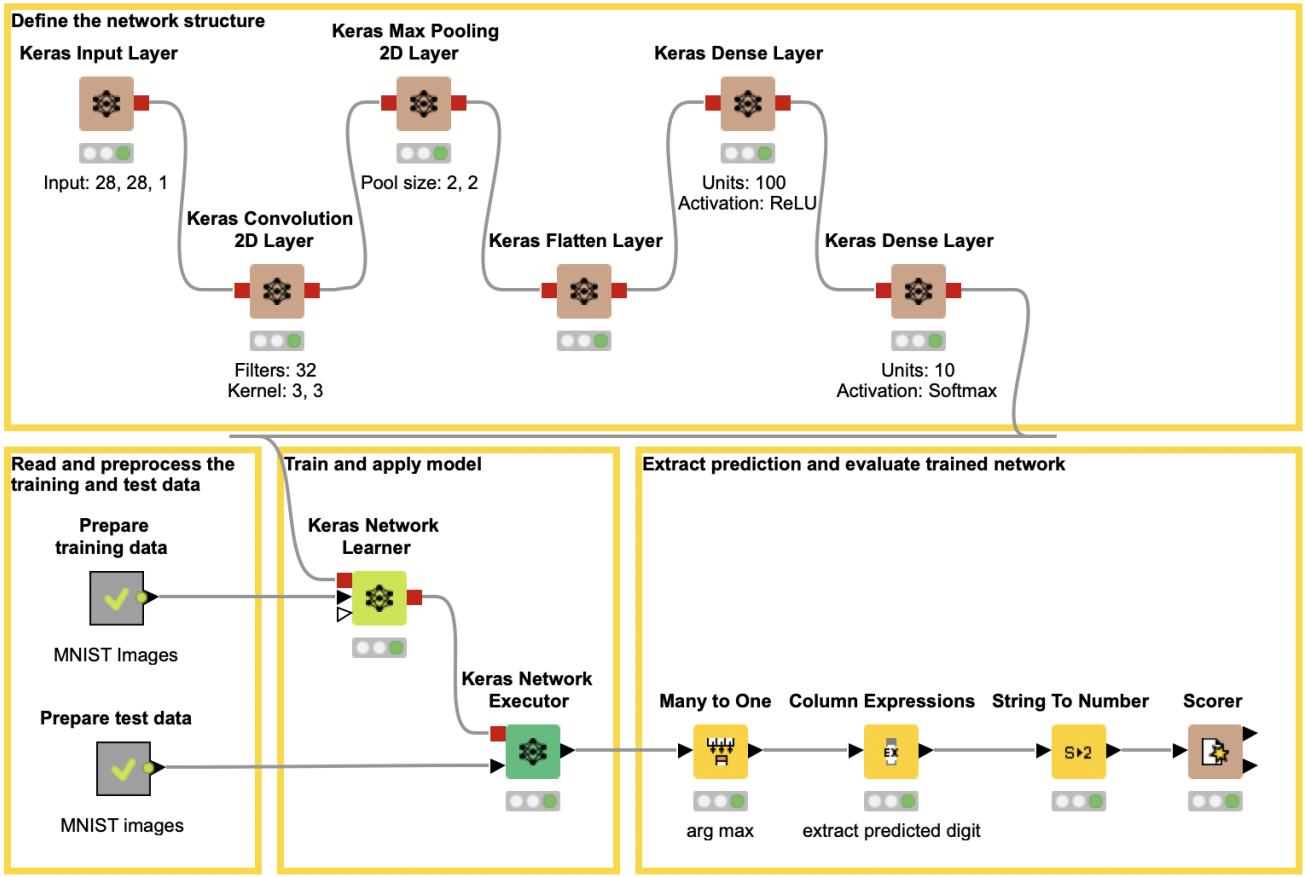
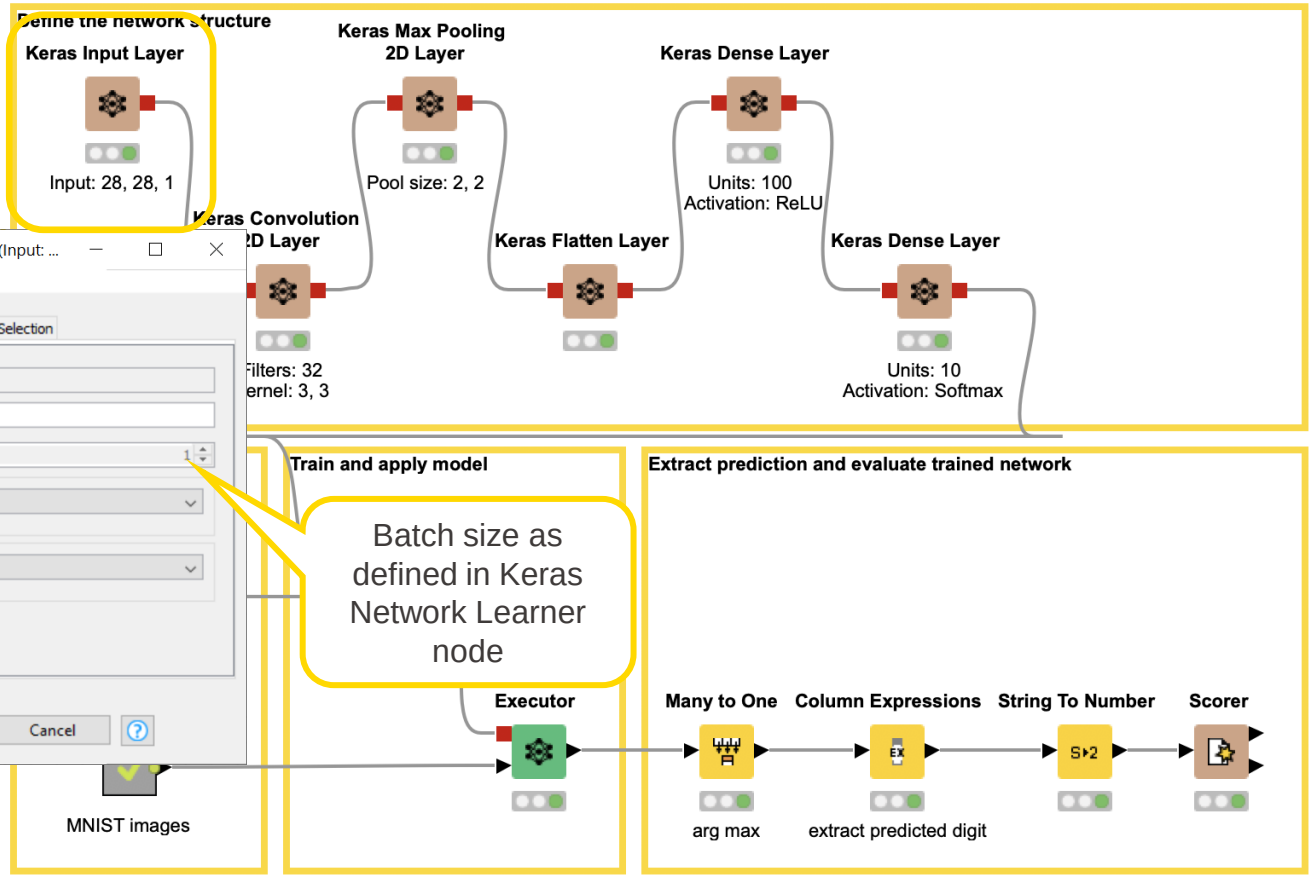


Image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

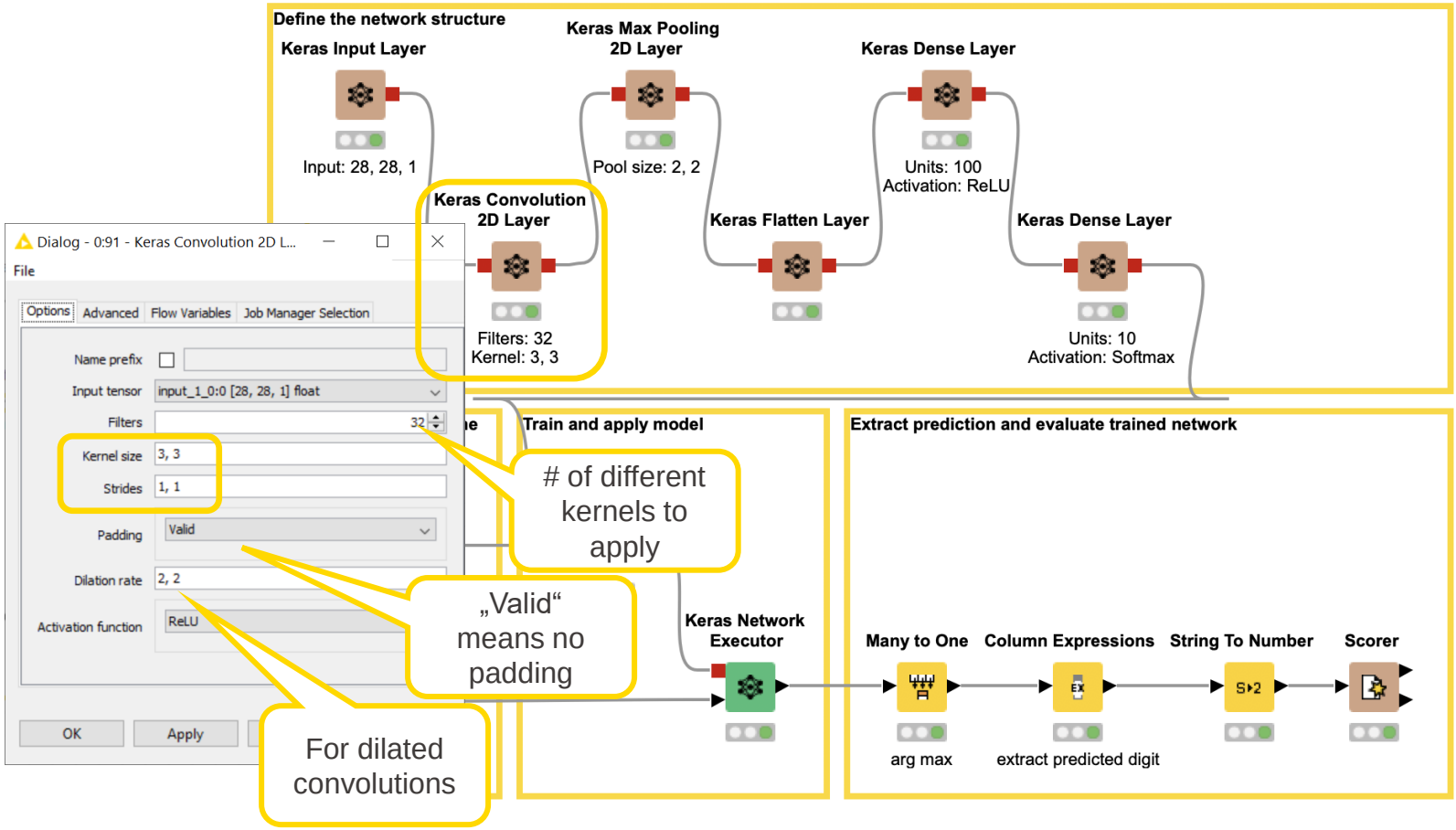
MNIST Classification with KNIME



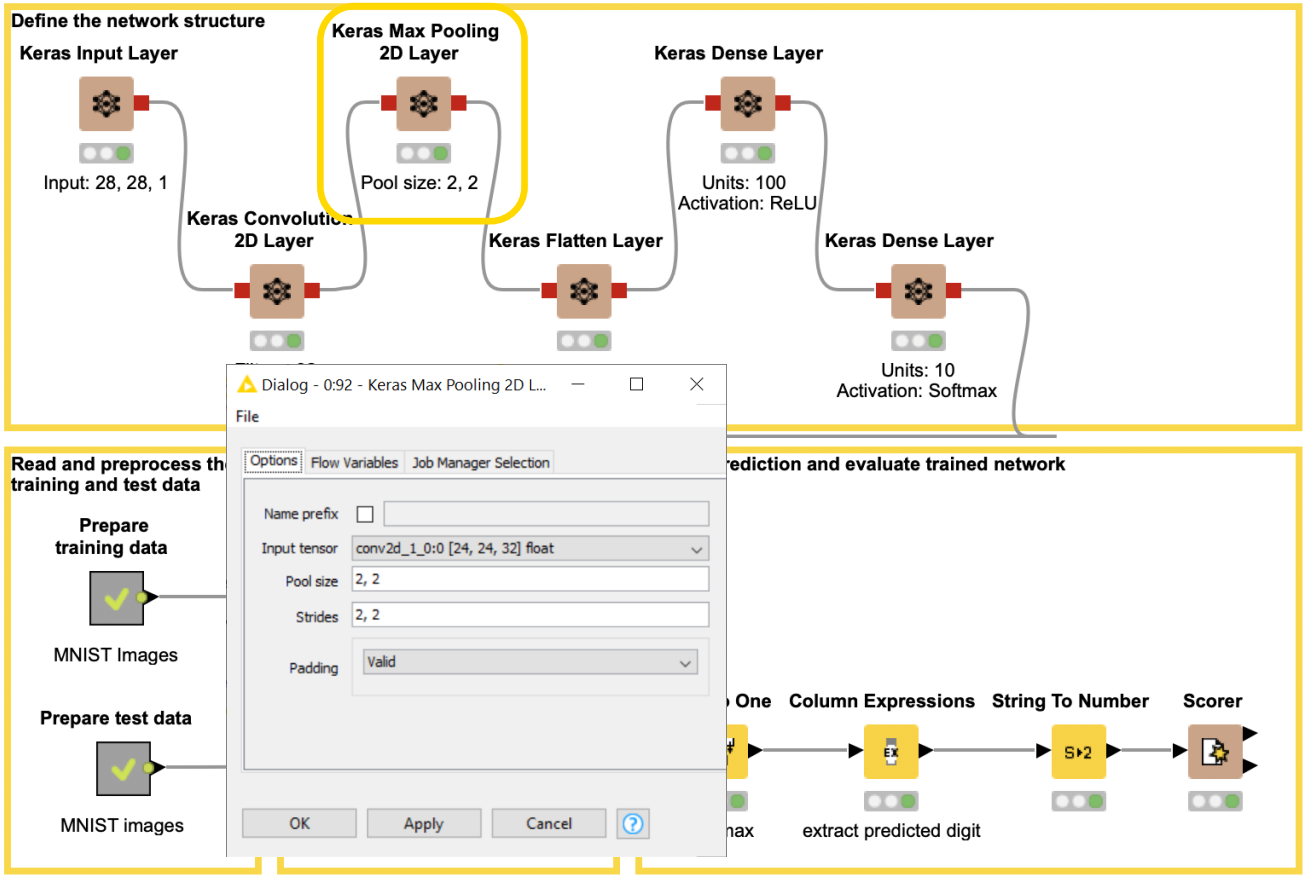
MNIST Classification with KNIME



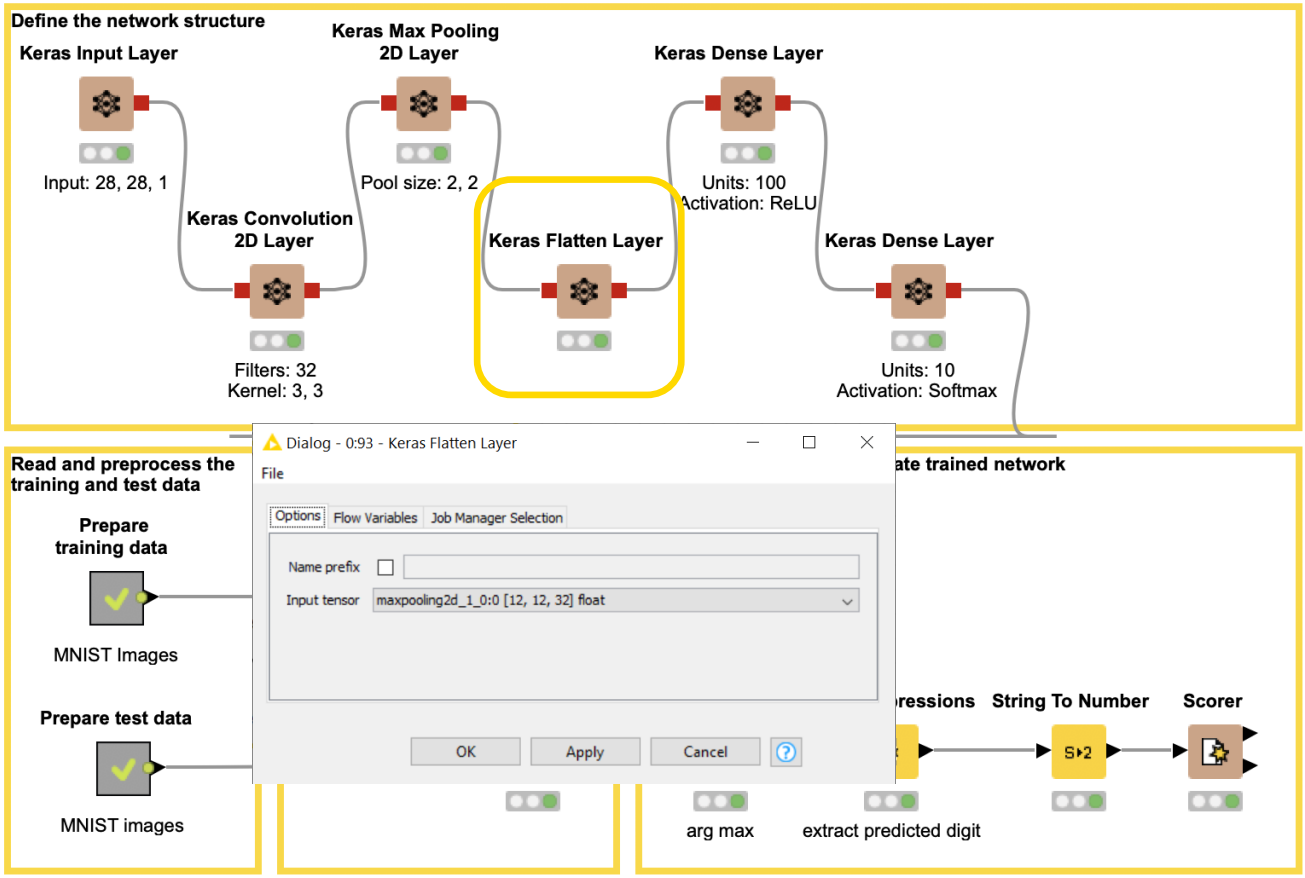
MNIST Classification with KNIME



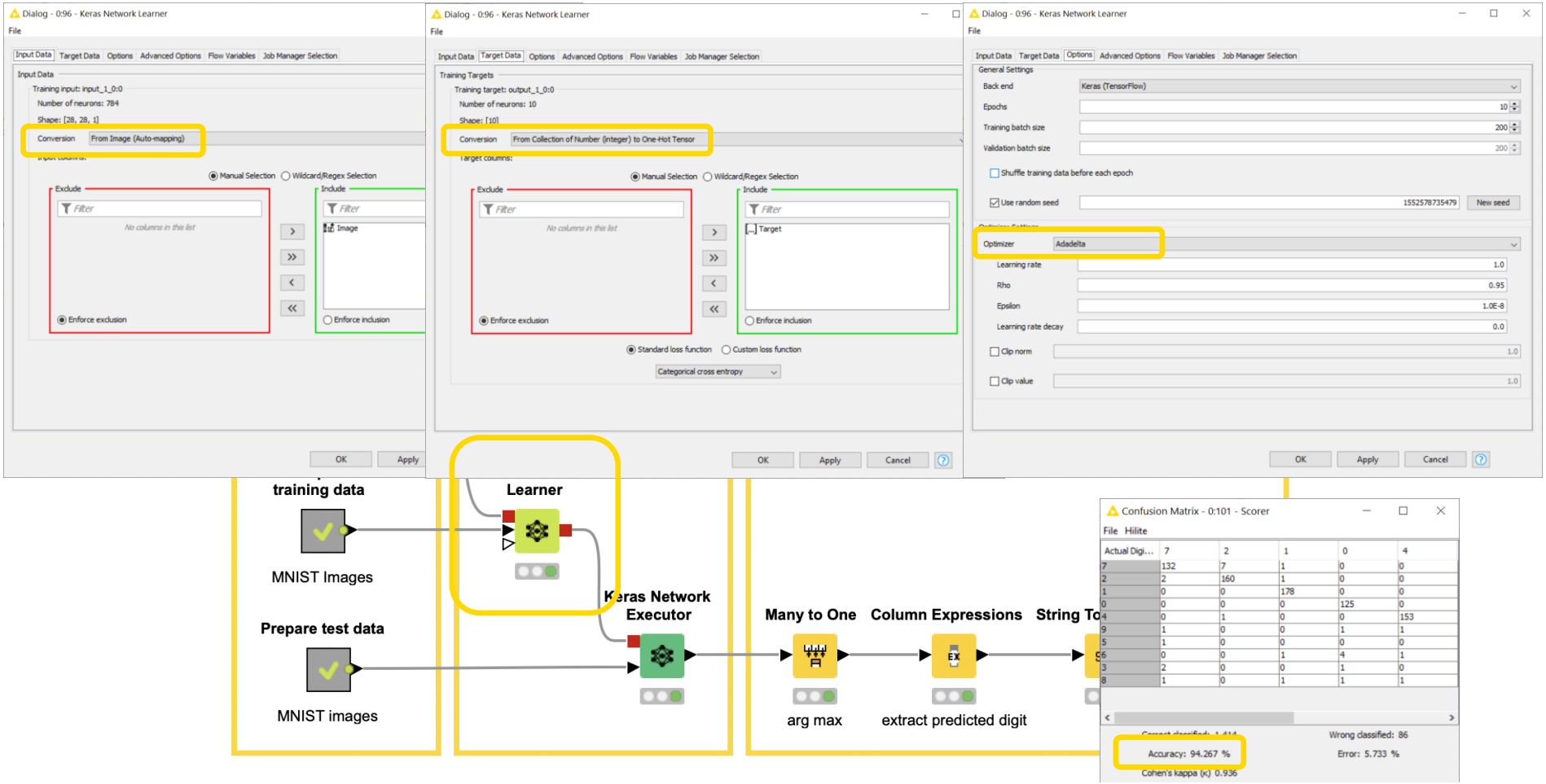
MNIST Classification with KNIME



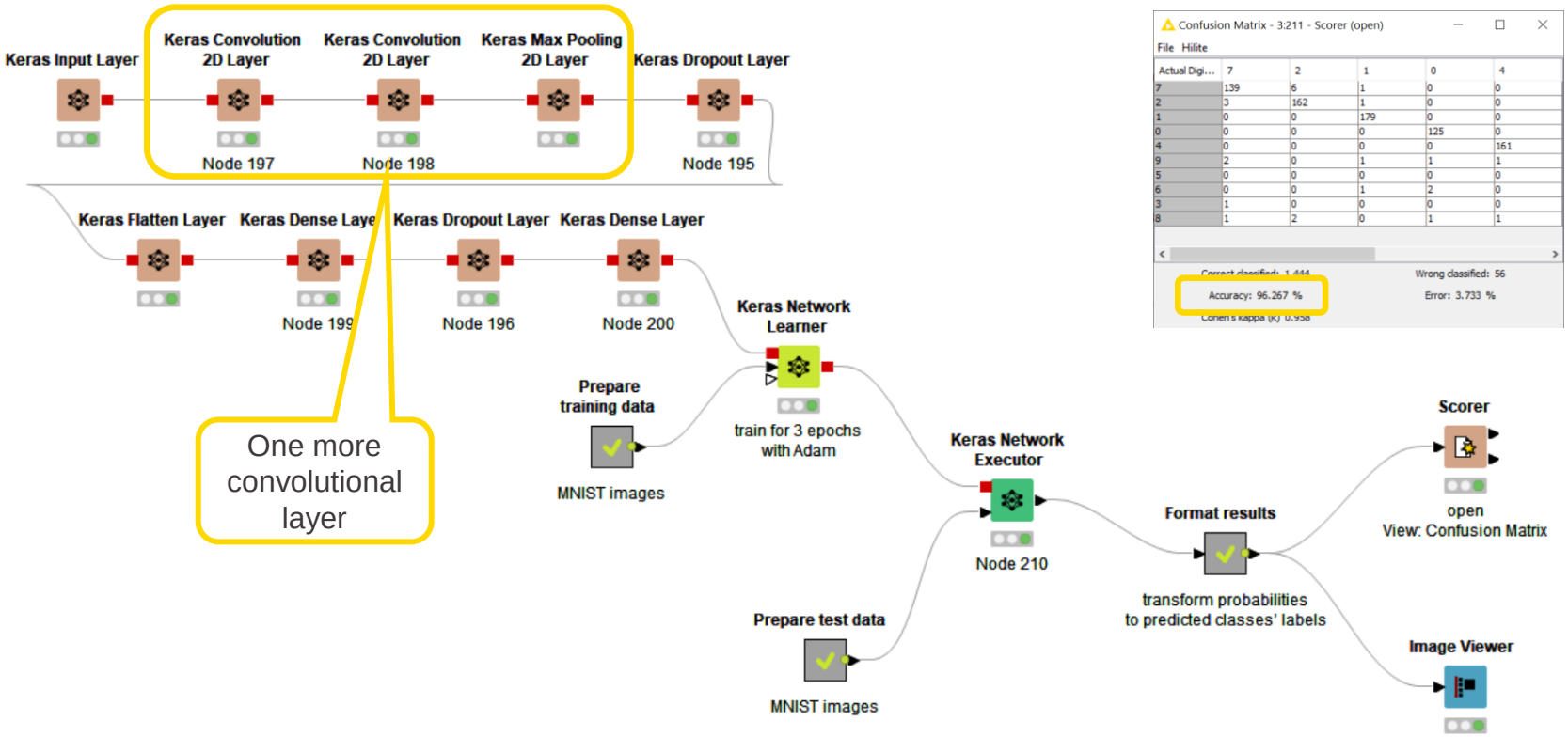
MNIST Classification with KNIME



MNIST Classification with KNIME



MNIST Classification with KNIME



CNN: Transfer Learning

- Training such networks is a long and complex process, requiring very powerful machines.
- Instead of retraining a new network completely from scratch, we could recycle existing networks, already built and trained by others on **similar** data.
- This technique is called ***Transfer Learning***.
- In Transfer Learning a model developed for a task is reused as the starting point for another model on a second task.
- On top of a previously trained network we add one or more neural layers
- We freeze all or some of the previously trained layers
- And we retrain only the remaining part of the whole network on our new task

Workflows on KNIME Hub

- Simple feed forward neural network

- MNIST Classification with one convolutional layer
  - Note: Please download the workflow group including the data folder. To do so you need to create an account and sing in.

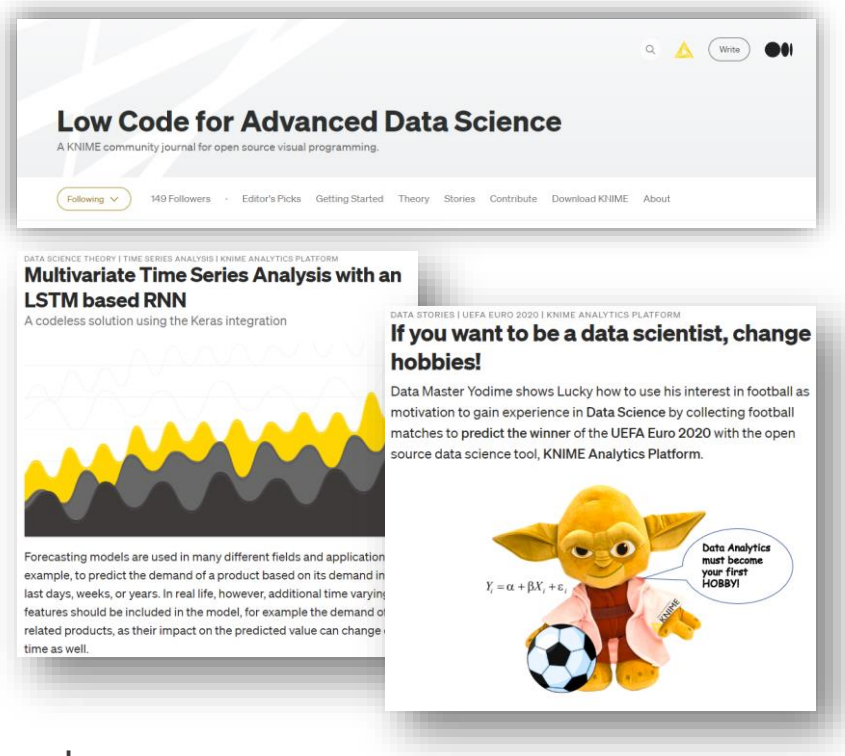
- MNIST Classification with two convolutional layers

Stay Up-To-Date and Contribute

- Follow the KNIME Community Journal on Medium
  [Low Code for Advanced Data Science]

- Daily content on data stories, data science theory, getting started with KNIME and more
  for the community by the community

- Would you like to share your data story
  with the KNIME community?

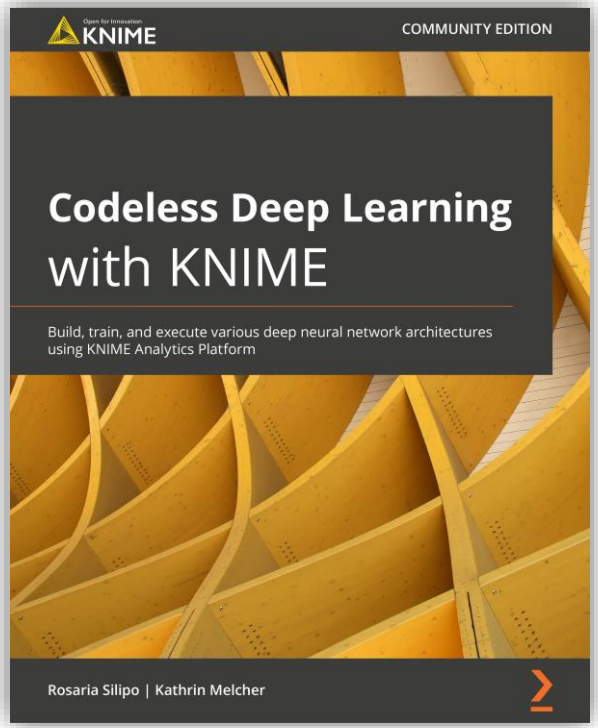


Contributions are always welcome!

Learn More!

- Codeless Deep Learning with KNIME—Packt, 2020
  - By Rosaria Silipo & Kathrin Melcher
  - Buy it here!

- [L4-DL] Introduction to Neural Networks and Deep Learning, July 11-15
  - Register here!



Thank you!