Open for Innovation
KNIME

Decoding Deep Learning

Demystified and Codeless with KNIME

Satoru Hayasaka, Ph.D.
satoru.hayasaka@knime.com

Data Scientist,
KNIME Inc., Austin, TX

This slide deck is available at
https://kni.me/s/4-i5-EcLrZwf5cxN

Agenda

- Introduction to deep learning
  - Machinery of deep learning with KNIME
  - Artificial neural networks
  - Back propagation
  - Optimizing neural network models
  - Demo – classification with a feedforward neural network (FFNN)
- Convolutional Neural Networks (CNN)
  - Computer vision
  - Convolution & pooling layers
  - Transfer learning
- Recurrent Neural Networks (RNN)
  - Sequential data & RNN
  - Long short-term memory (LSTM)

Target Audience

- Those interested in deep learning
  - With some background knowledge in machine learning
- No KNIME experience? No problem!
  - General introduction to deep learning
  - Some pointers to get started with DL with KNIME
  - Links to workflow examples on KNIME Hub

What is Deep Learning?

Artificial intelligence

Any technique that enables machines to mimic human intelligence

Machine learning

Ability to learn without being explicitly programmed using past observations

Artificial neural networks

Extract patterns using neural networks

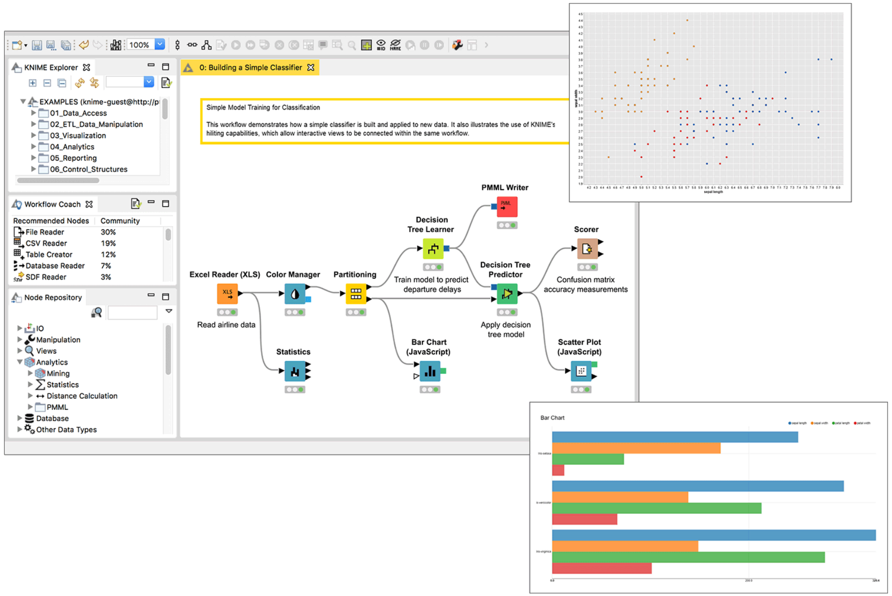
Deep learning

Modern revolution of neural networks

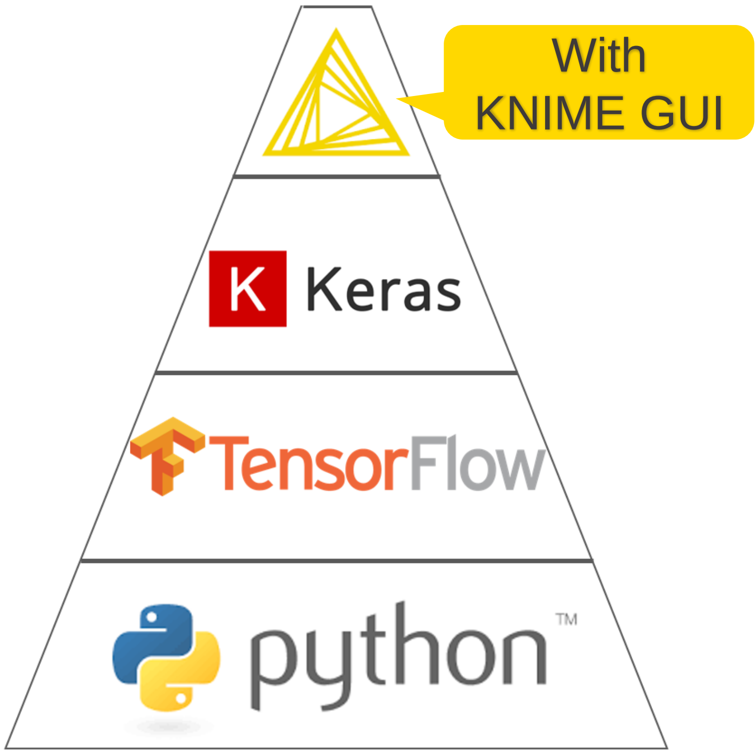
Machinery of deep learning with KNIME

What is KNIME Analytics Platform?

- A tool for data analysis, manipulation, visualization, and reporting
- Based on the graphical programming paradigm
- Provides a diverse array of extensions:
  - Text Mining
  - Network Mining
  - Cheminformatics
  - Many integrations,
    such as Java, R, Python,
    Weka, Keras, Plotly, H2O, etc.



Keras + KNIME



- KNIME Deep Learning Extension builds on top of the Keras Libraries
- The Keras libraries build on top of TensorFlow
- Deep Learning libraries from TensorFlow and Keras are accessible via Python ...
- ... And KNIME with the Deep Learning Keras Integration.

Installation

Deep Learning in KNIME Analytics Platform comes with a specific integration.
A few simple steps are necessary to get it up and running.

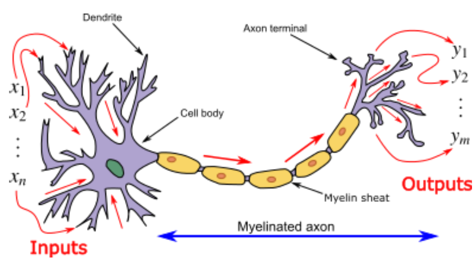
- On your machine:
  - **Anaconda** with Python3 correctly installed
- Extensions installed on KNIME Analytics Platform
  - **KNIME Deep Learning - Keras Integration**
  - **KNIME Deep Learning - TensorFlow Integration**

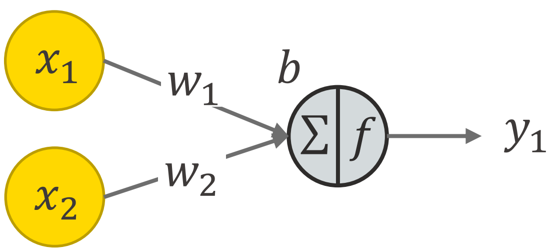
NOTE: This is just a quick start guide to start using Deep Learning with your KNIME Analytics Platform.
If you are experiencing issues or want to customize your installation, please refer to
KNIME Deep Learning Integration Installation Guide

Artificial neural networks

Let’s Start Simple

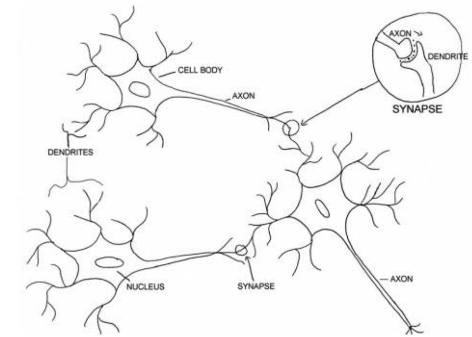
Single neuron

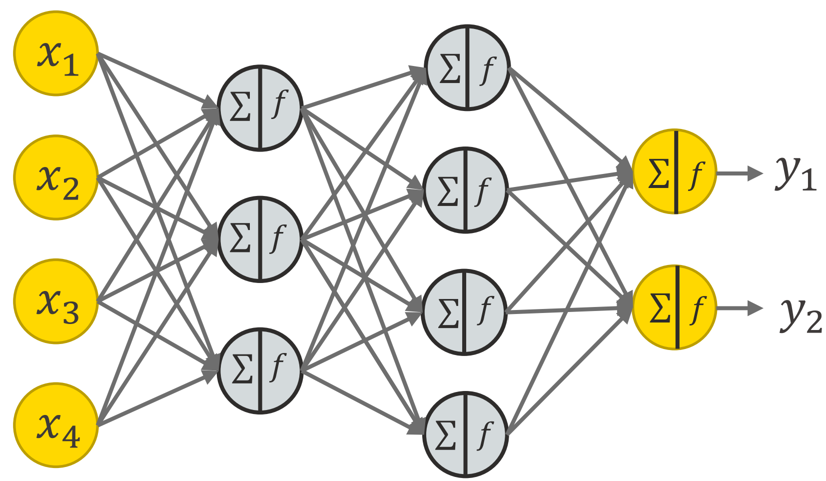




$$x_1 w_1 + x_2 w_2 + b$$

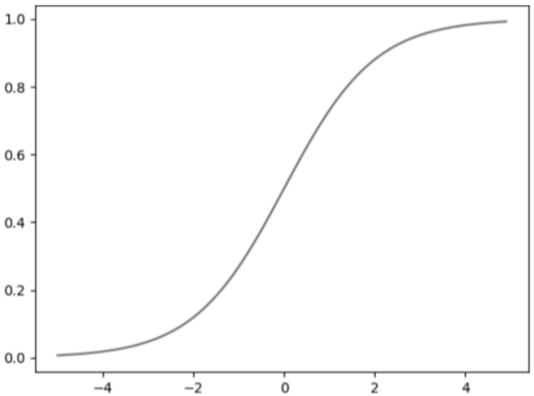
Neural network





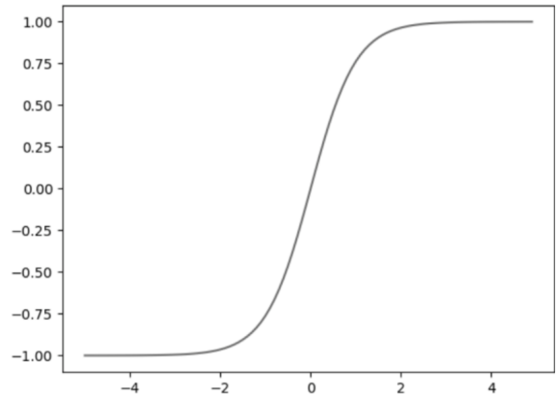
Frequently used Activation Functions

Sigmoid



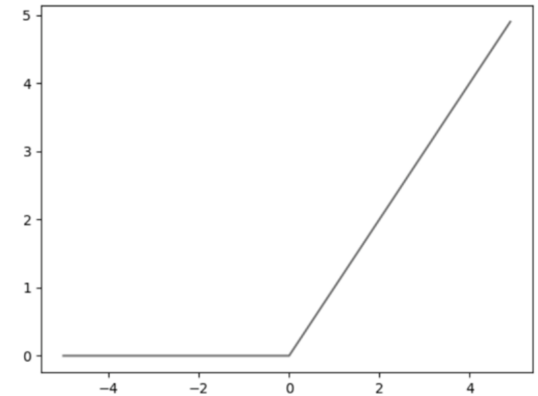
$$f(a) = \frac{1}{1 + e^{-a}}$$

Tanh



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

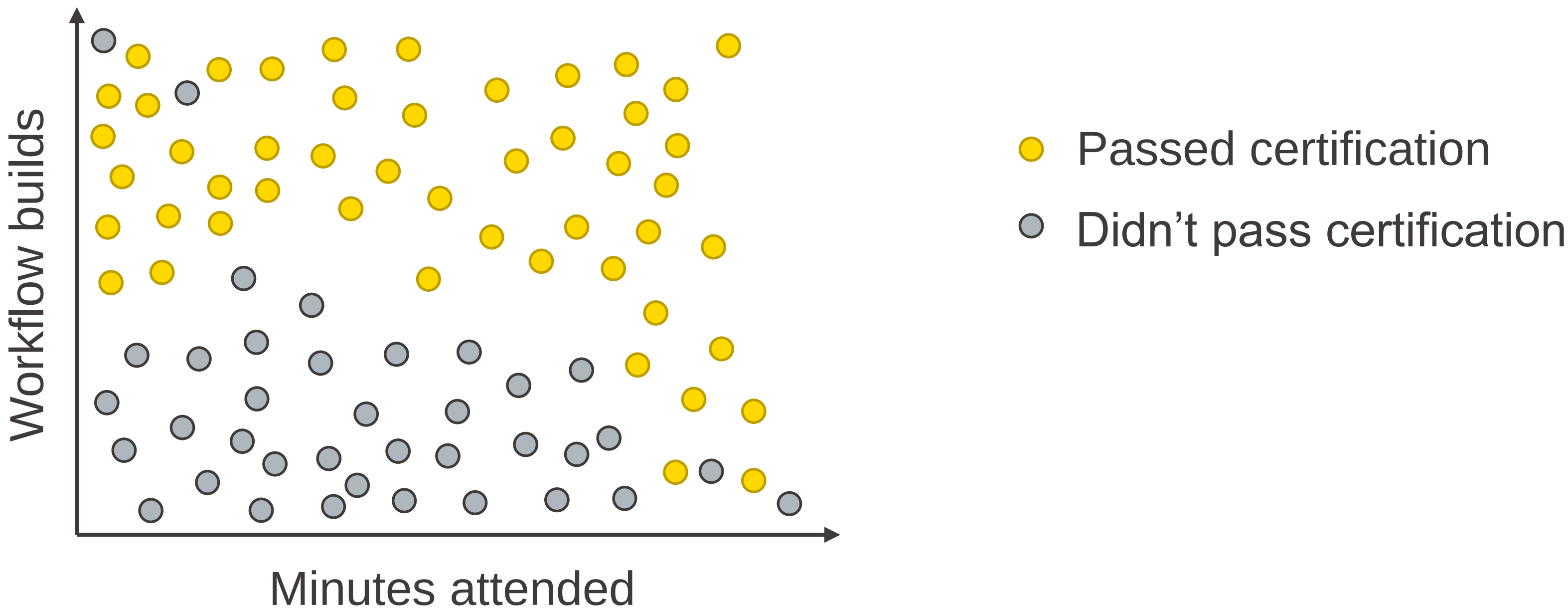
Rectified Linear Unit (ReLU)



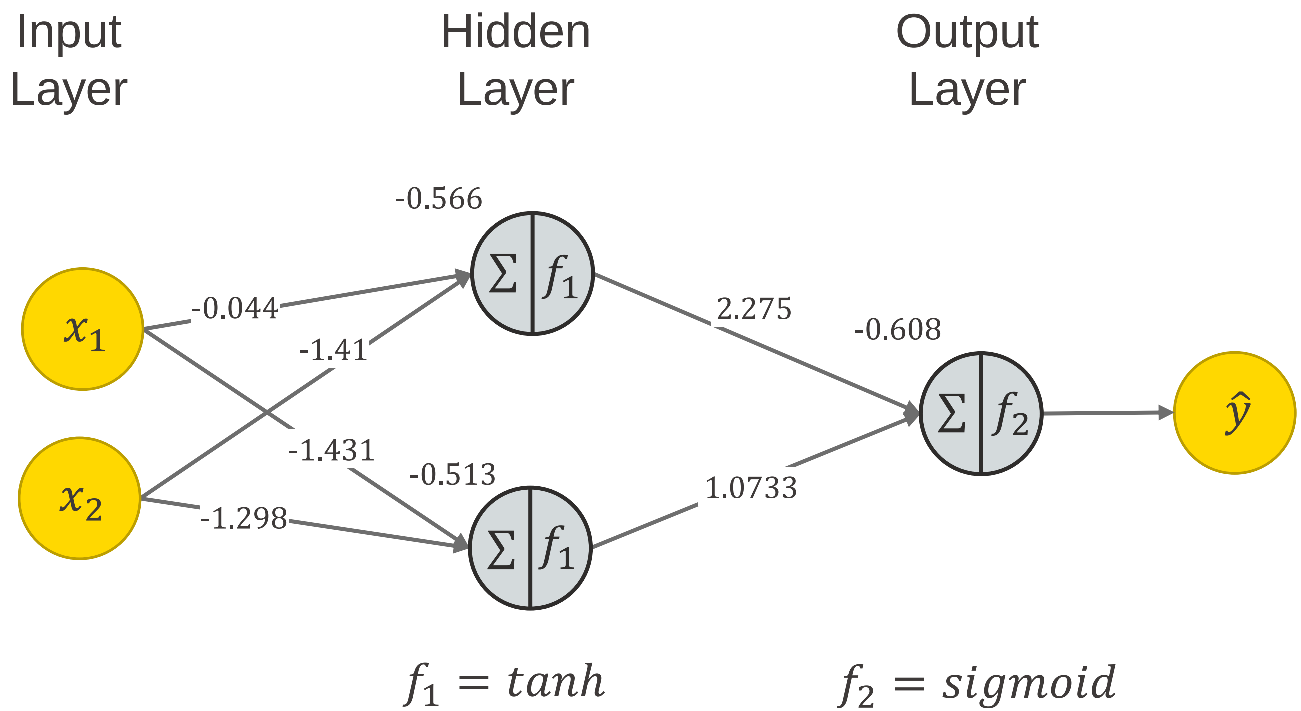
$$f(a) = max\{0, a\}$$

Non-linear activation functions enable modeling of non-linear problems

Example: Passing the KNIME L1-Certification



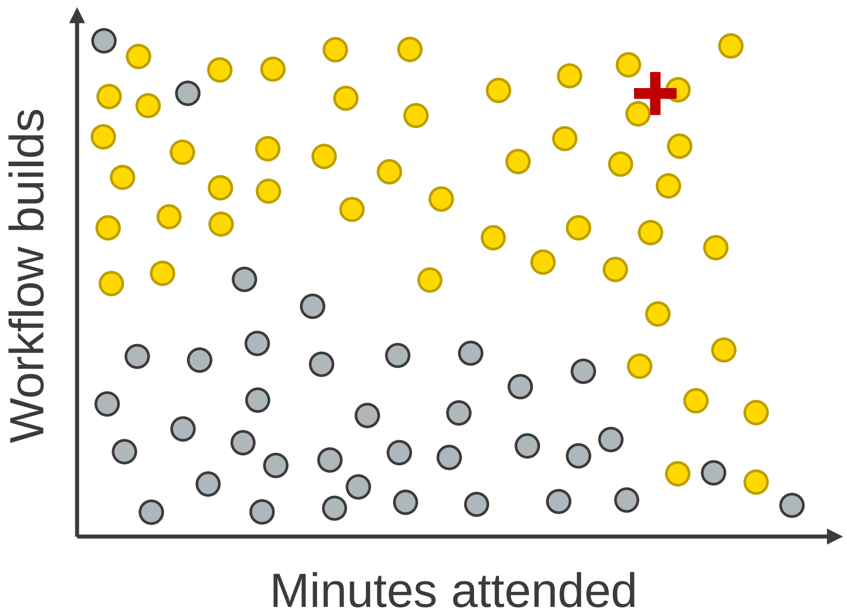
Example: Passing the KNIME L1-Certification



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



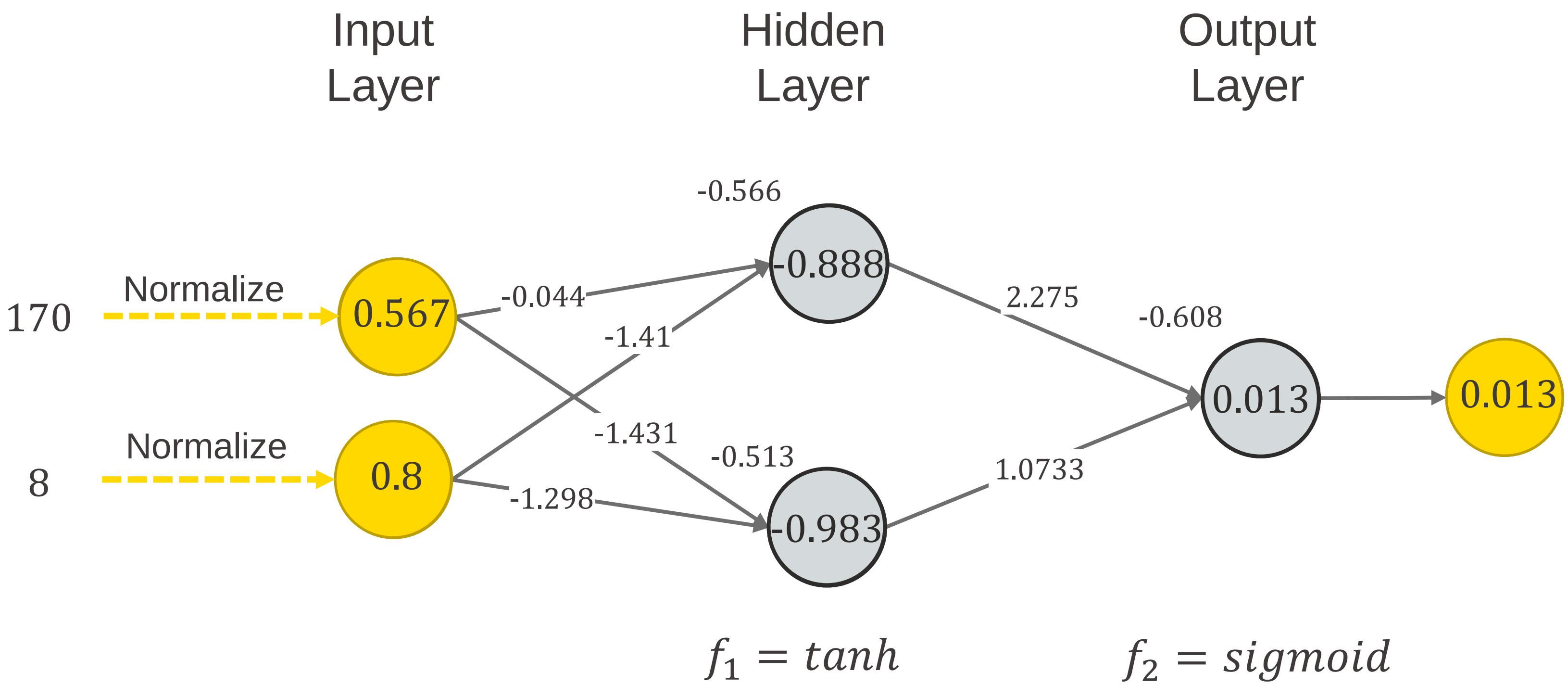
Passed certification

Didn’t pass certification

New sample

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

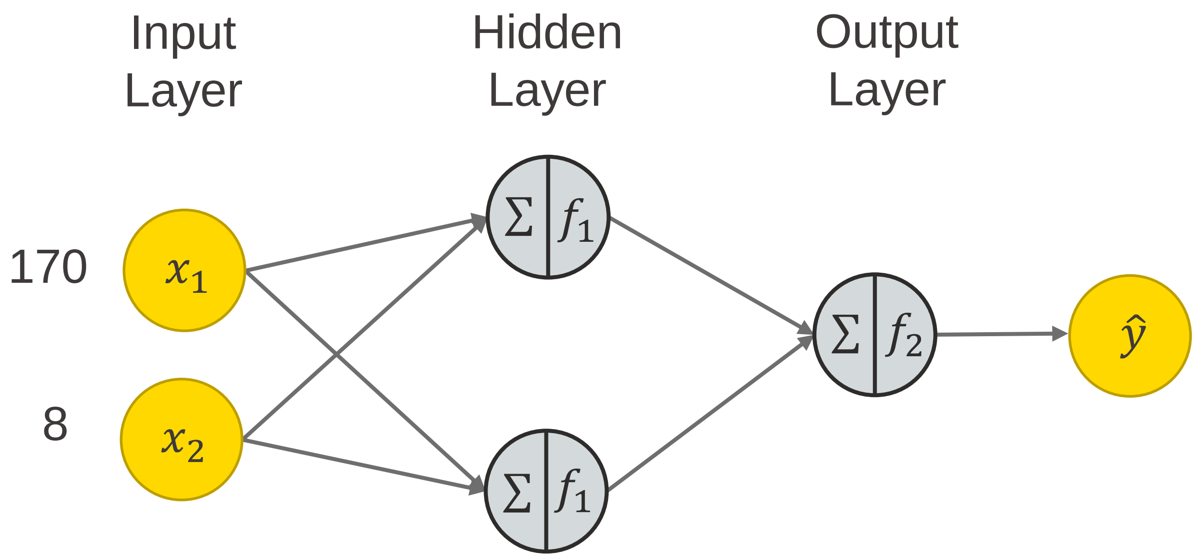
Example: Passing the KNIME L1-Certification



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



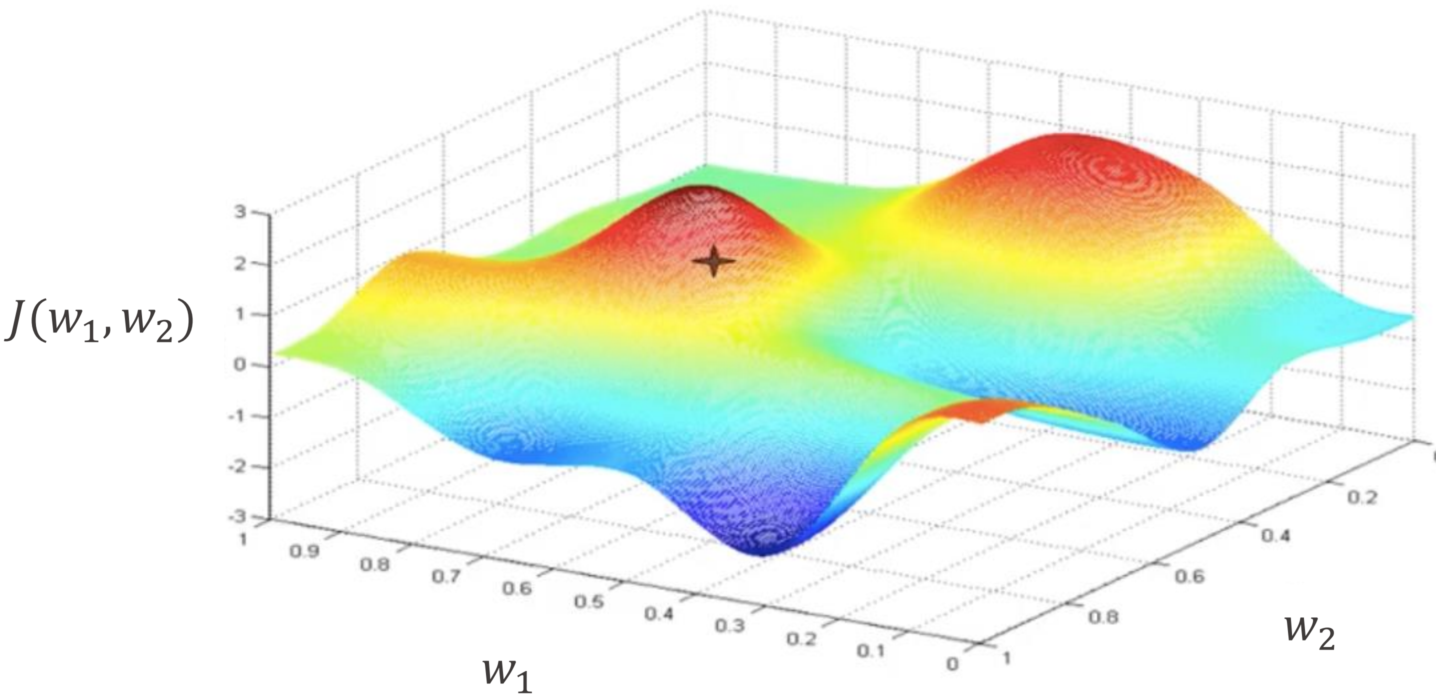
Predicted $\hat{y}$ = 0.013
True $y = 1$

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



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

Learning Rule from Gradient Descent

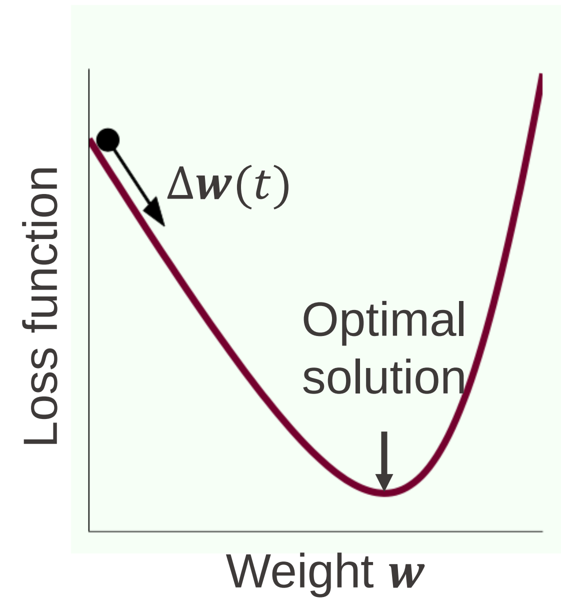
- Adjust the weight for the next step by the adjustment term $\Delta \boldsymbol{w}(t)$

$$\boxed{\boldsymbol{w}(t+1)} = \boxed{\boldsymbol{w}(t)} + \boxed{\Delta \boldsymbol{w}(t)}$$

Weight in current step t

Weight adjustment based on gradient

Updated weight in next step $t+1$



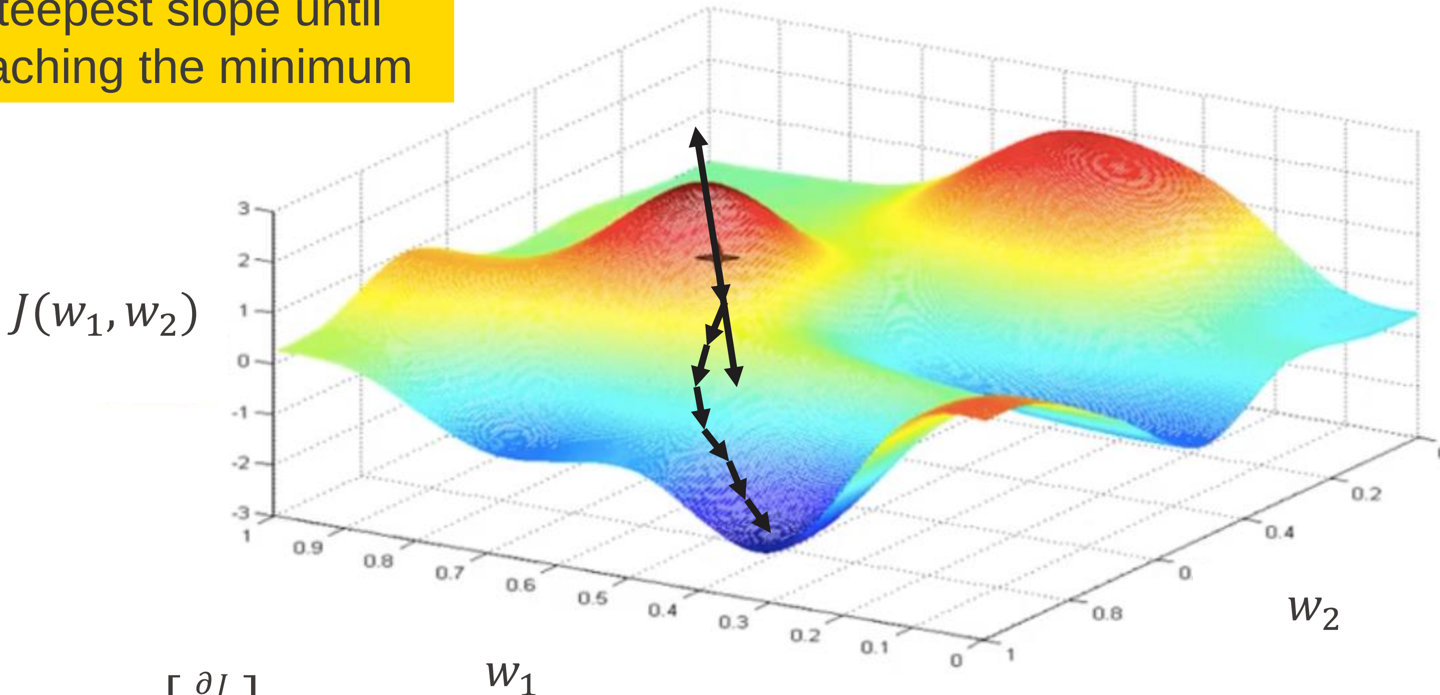
Idea Behind Gradient Descent

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

Rolling down the steepest slope until reaching the minimum



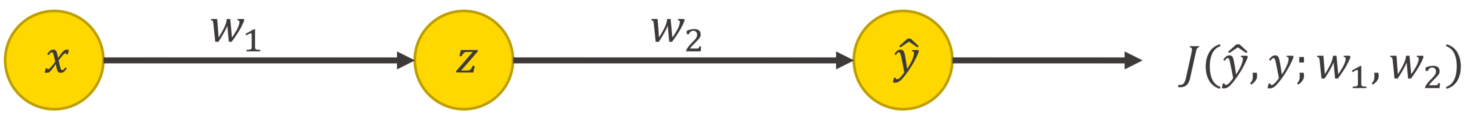
$$\nabla_W J(x, W) = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \end{bmatrix}$$

Back propagation

Backpropagation

- Efficient way to calculate the gradient during optimization

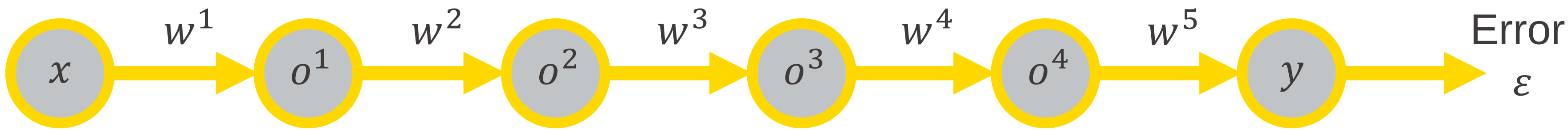
Forward pass



Backward pass

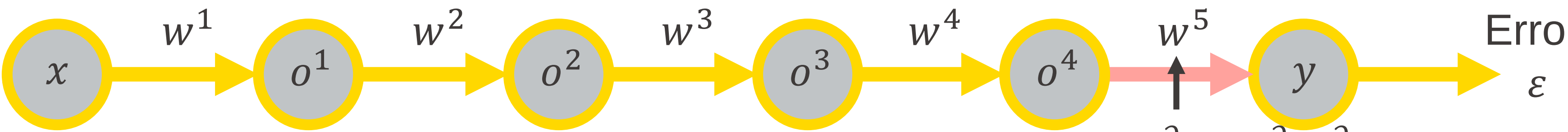
Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule



Gradients by Chain Rule

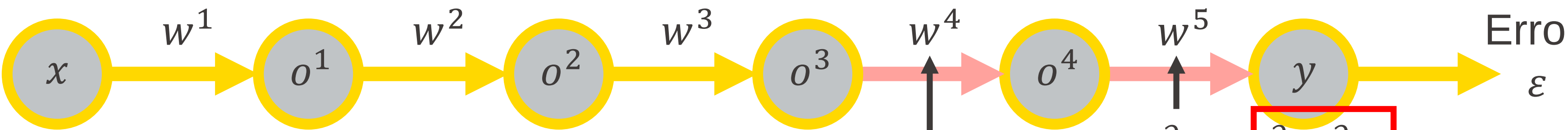
- Gradients can be determined – one layer at a time – by the chain rule



$$\frac{\partial \varepsilon}{\partial w^5} = \frac{\partial \varepsilon}{\partial y} \frac{\partial y}{\partial w^5}$$

Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule

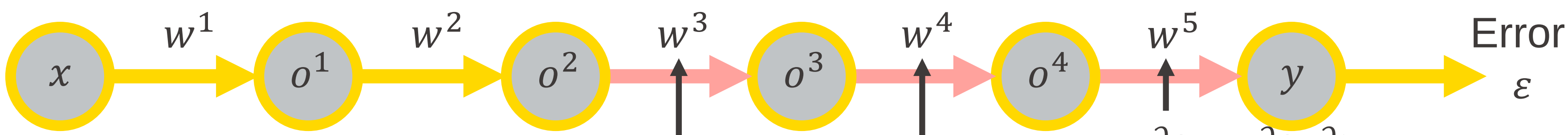


$$\frac{\partial \varepsilon}{\partial w^5} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}}$$

$$\frac{\partial \varepsilon}{\partial w^4} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}$$

Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule



$$\frac{\partial \varepsilon}{\partial w^5} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}$$

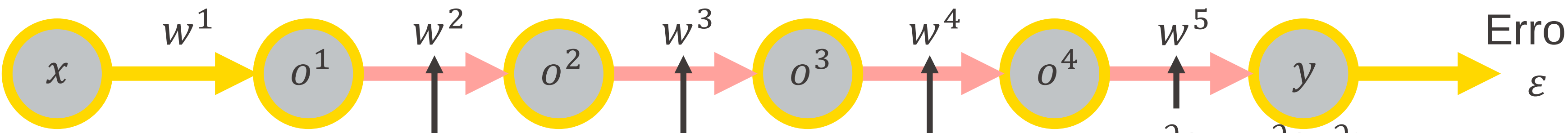
$$\frac{\partial \varepsilon}{\partial w^4} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}}$$

$$\frac{\partial \varepsilon}{\partial w^3} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}$$

Open for Innovation
KNIME

Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule



$$\frac{\partial \varepsilon}{\partial w^5} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}$$

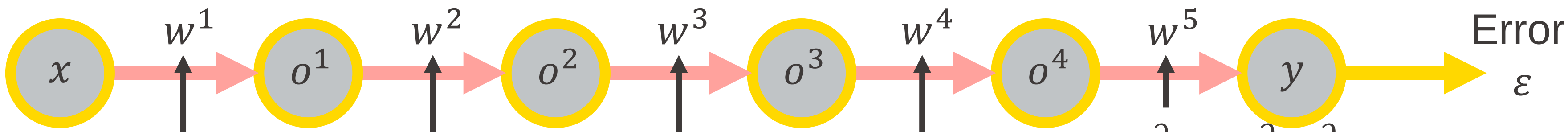
$$\frac{\partial \varepsilon}{\partial w^4} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}$$

$$\frac{\partial \varepsilon}{\partial w^3} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}}$$

$$\frac{\partial \varepsilon}{\partial w^2} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}}\frac{\partial w^3}{\partial o^2}\frac{\partial o^2}{\partial w^2}$$

Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule



$$\frac{\partial \varepsilon}{\partial w^5} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}$$

$$\frac{\partial \varepsilon}{\partial w^4} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}$$

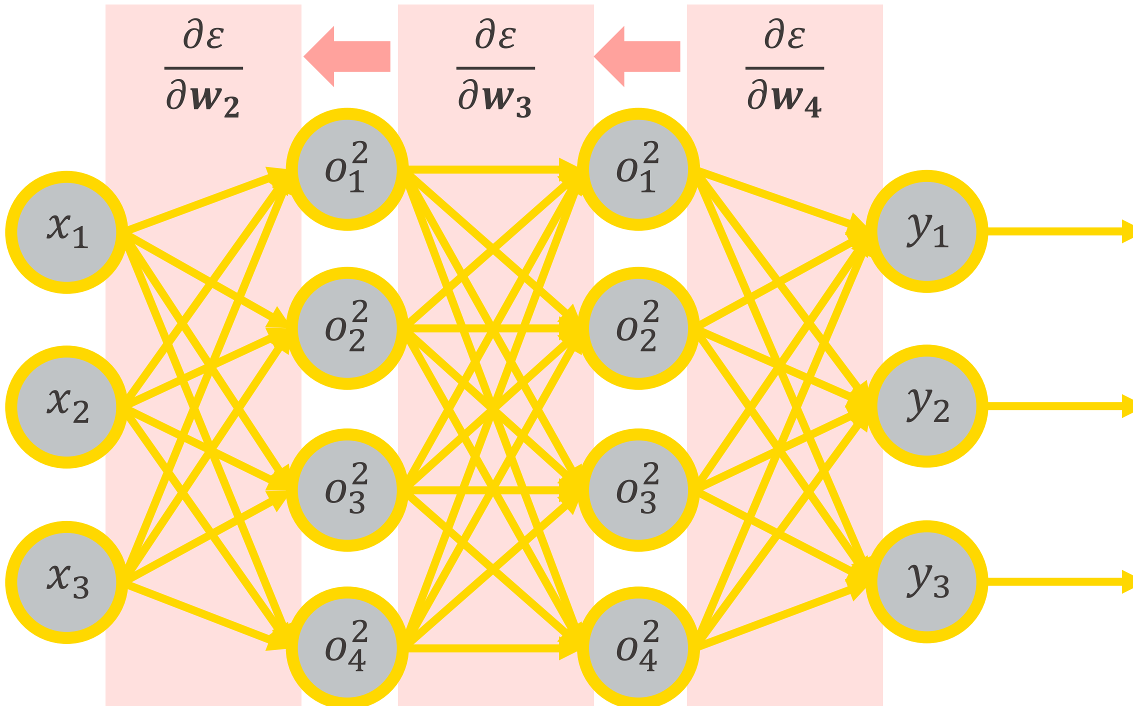
$$\frac{\partial \varepsilon}{\partial w^3} = \frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}$$

$$\frac{\partial \varepsilon}{\partial w^2} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}\frac{\partial w^3}{\partial o^2}\frac{\partial o^2}{\partial w^2}}$$

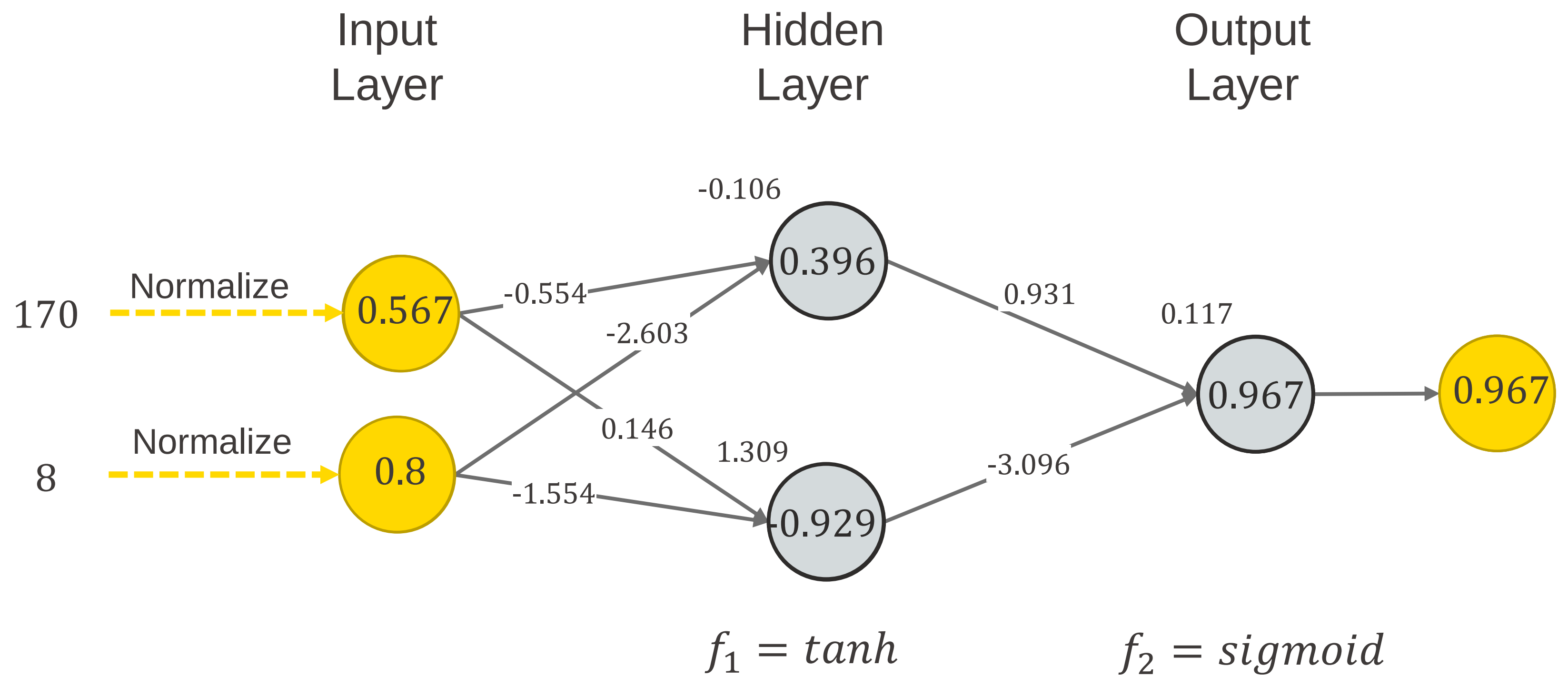
$$\frac{\partial \varepsilon}{\partial w^1} = \boxed{\frac{\partial \varepsilon}{\partial y}\frac{\partial y}{\partial w^5}\frac{\partial w^5}{\partial o^4}\frac{\partial o^4}{\partial w^4}\frac{\partial w^4}{\partial o^3}\frac{\partial o^3}{\partial w^3}\frac{\partial w^3}{\partial o^2}\frac{\partial o^2}{\partial w^2}}\frac{\partial w^2}{\partial o^1}\frac{\partial o^1}{\partial w^1}$$

Gradients by Chain Rule

- Gradients can be determined – one layer at a time – by the chain rule
- To determine the gradient at a particular layer, you only need gradients from the subsequent layers ➔ known as ***back-propagation***



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 $y < 0.5 \Rightarrow Failed$

Optimizing neural network models

Loss Functions

- Quantifies errors or deviations in the network outcome compared to the target
- *We want to minimize the loss!!*
- Different types of loss functions for classification and regression

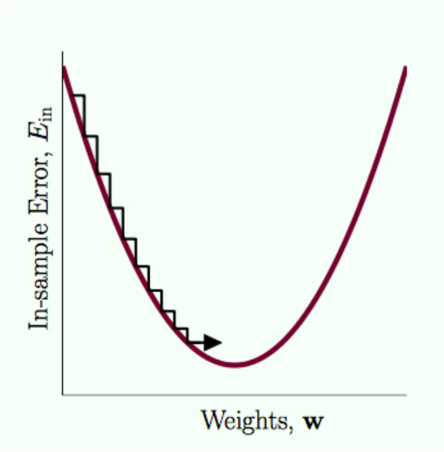
- *Classification*: We want the predicted category to match the target
- *Regression*: We want to minimize deviation from the target

Different Loss Functions

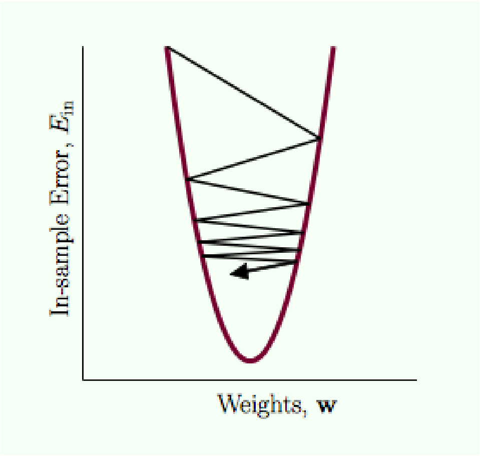
- Binary classification
  - Binary cross entropy
- Multi-class classification
  - Categorical cross entropy
- Regression problem
  - Mean squared error (MSE)
  - Mean absolute error (MAE)

Learning Rate η

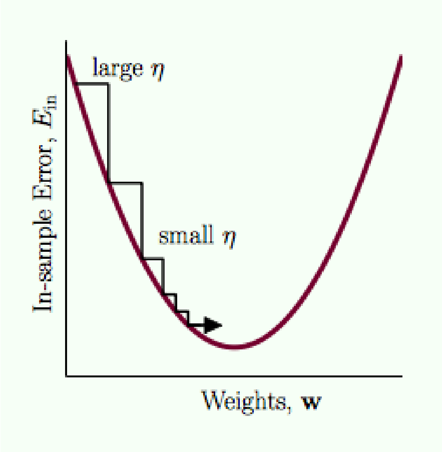
η too small



η too large



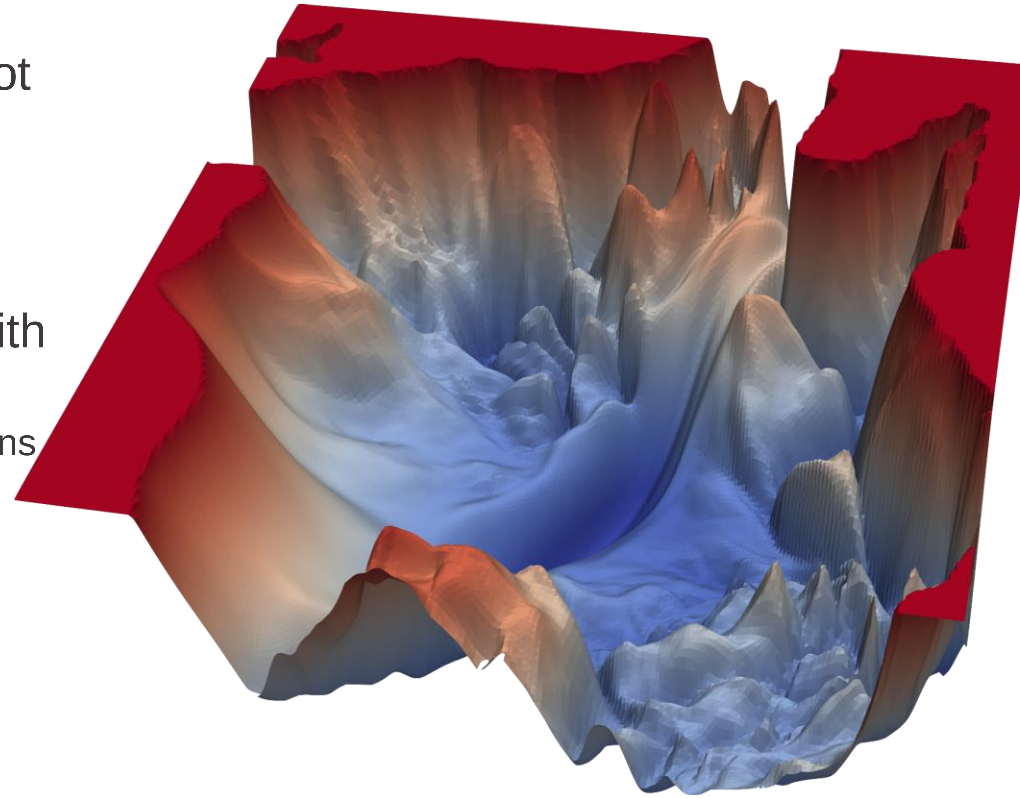
η just right



Loss Landscape of a Real Neural Network

- In reality, loss landscape may not be smooth
  - Possibly many local minima

- Different optimizer algorithms with
  - Varying learning rate η
  - History of gradients in previous iterations



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

Optimizers in Keras

Optimizer	How it works	Strengths	Weaknesses	When to use
SGD with momentum	Use the previous gradient to accelerate convergence	-Reduces oscillation near maxima	-Const learning rate	
NAG (Nesterov accelerated gradient)	Use the current gradient to predict gradient	-Increased responsiveness	-Additional hyperparameter	RNN
Adagrad	Updating by cumulating sum of sq gradients from past	-Different learning parameters for different features	-Computationally expensive -Shrinking learning rate	Sparse data (e.g. text)
Adadelta	Modified Adagrad with decaying average of sq gradients from past	-Learning rate not dramatically shrinking like Adagrad	-Computationally expensive	Sparse data (e.g. text)
RMSProp	Modified Adagrad with sq gradients added very slowly	-Learning rate not dramatically shrinking like Adagrad		
Adam (Adaptive Moment Estimation)	RMSProp plus decaying average of gradients from past	-Fast convergence	-Computationally expensive	

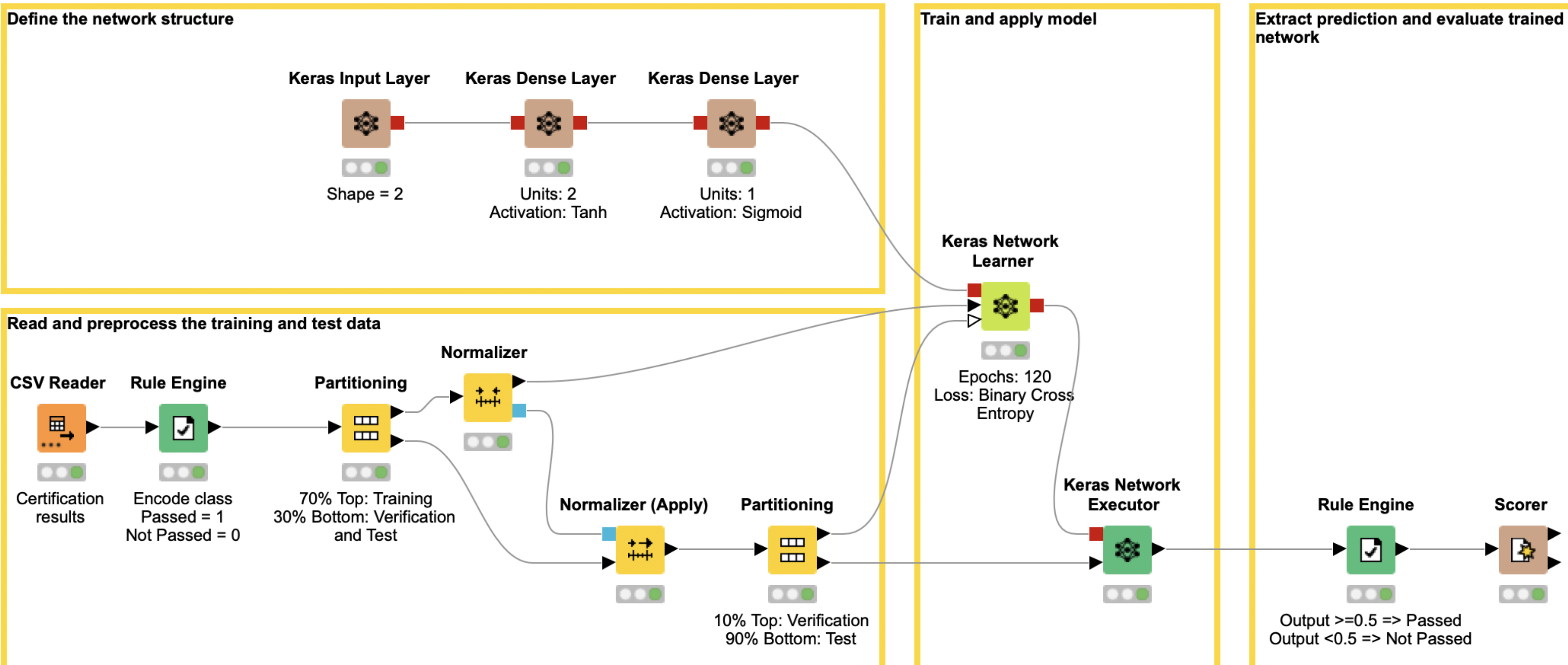
Which Activation Functions? Which Loss Functions?

- Depends on the problem you are working on

✓ Recommended
Δ Can be used

		Activation Functions								Loss Functions					
		Hidden Layers			Output Layer										
	Problems	Sigmoid	Tanh	ReLU	Sigmoid	Tanh	Linear	ReLU	Softmax	Binary CE	Hinge	Categorical CE	MSE	MSLE	MAE
Classification	Binary classification (0 vs 1)	✓	✓	✓	✓					✓					
	Binary classification (-1 vs 1)	✓	✓	✓		✓					✓				
	Multi-class classification	✓	✓	✓					✓			✓			
Regression	Regression	✓	✓	✓	Δ	Δ	✓	Δ					✓		
	Regression (wide range)	✓	✓	✓			✓							✓	
	Regression (possible outliers)	✓	✓	✓			✓								✓

Codeless Deep Learning with KNIME Analytics Platform



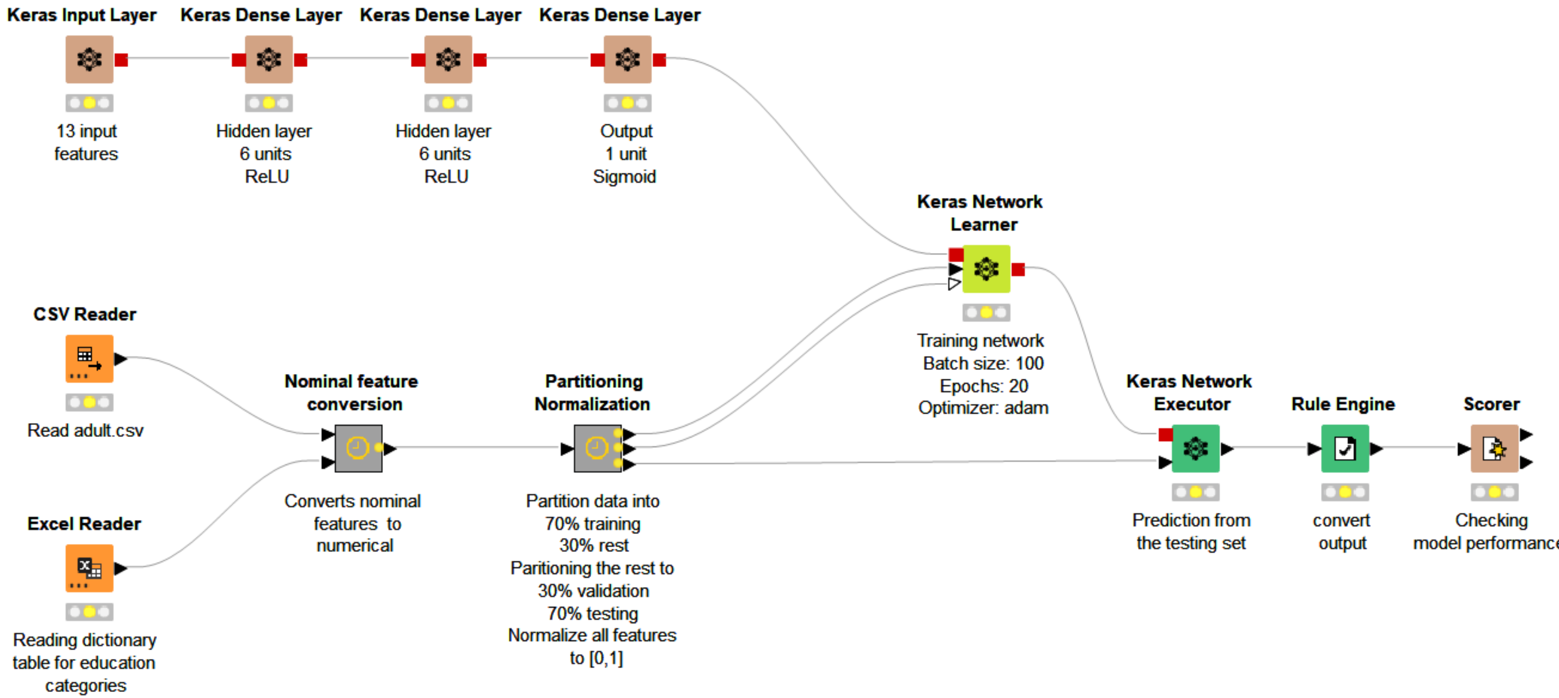
Download the workflow from the KNIME Hub

Demo – adult data classifier

- Adult data set: demographic data of 32k adults
- Goal: Binary classification whether the income is above $50k
- 13 features – numerical and nominal
- Train an ANN with 13-6-6-1 units

- ➔ Demo with KNIME Analytics Platform

Demo – adult data classifier



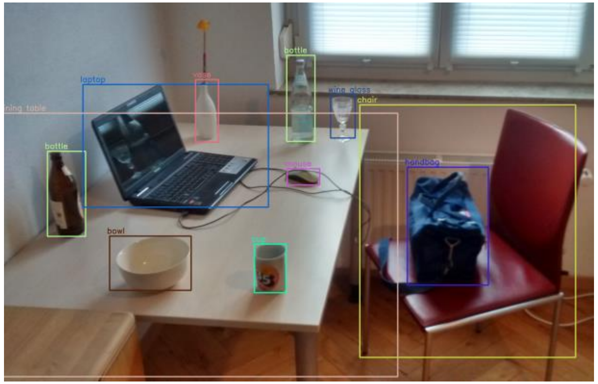
Download the workflow from the KNIME Hub

Computer vision:
Challenges working with image data

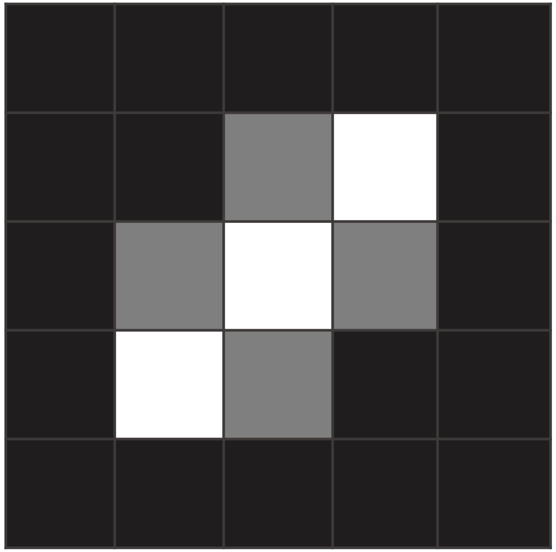
Why is Computer Vision Important?

- Increasing amount of video and image data
  - 30 000 minutes of video are uploaded to YouTube every minute

- Many application areas and use cases:
  - Image classification / image recognition
    - Detecting of diseases
    - Detecting of anomalies
    - Face recognition to unlock a phone or door
  - Object detection
    - Marking objects in an image, e.g., traffic signs
  - Semantic segmentation
  - Neural style transfer



How Can We Represent a Gray-Scale Image?



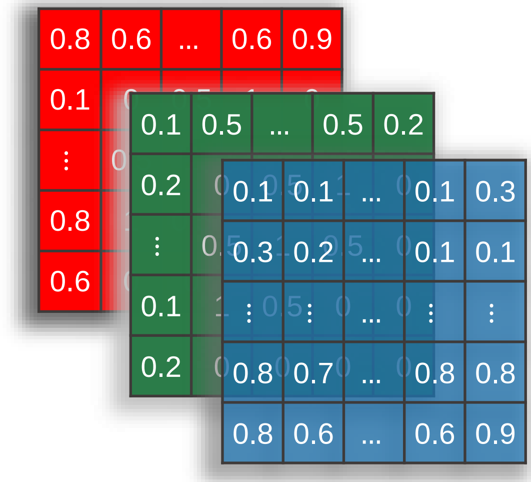
=

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?

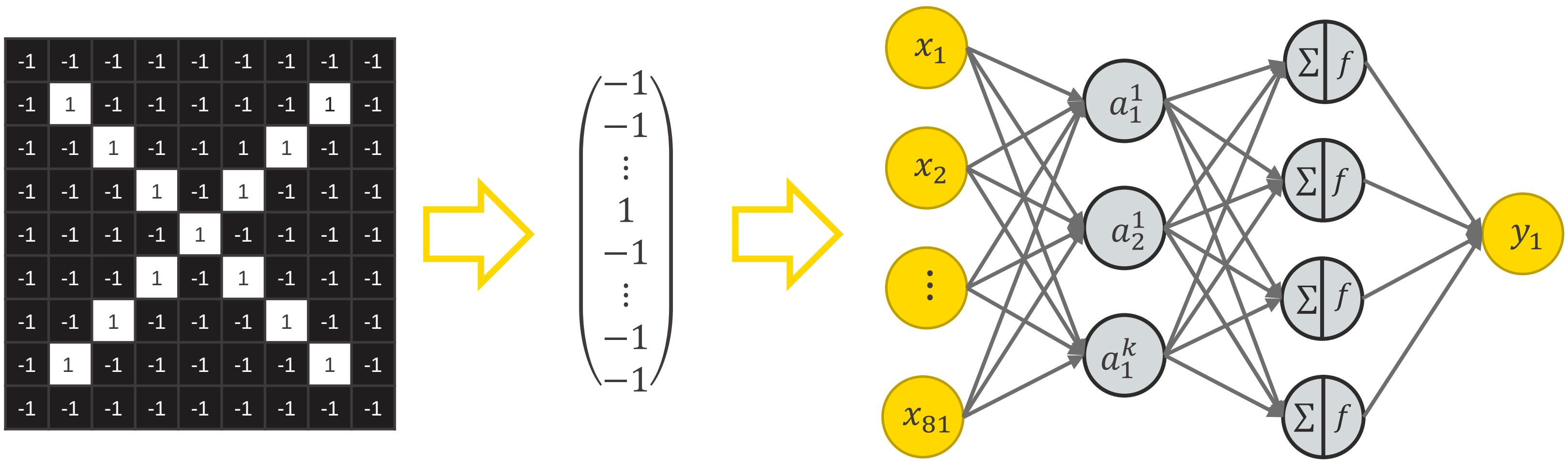
=



- 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.

Problems with FFNN for Image Classification

- Goal: Train network to recognize x’s
- Approach: Flatten the image and apply a feed forward neural network



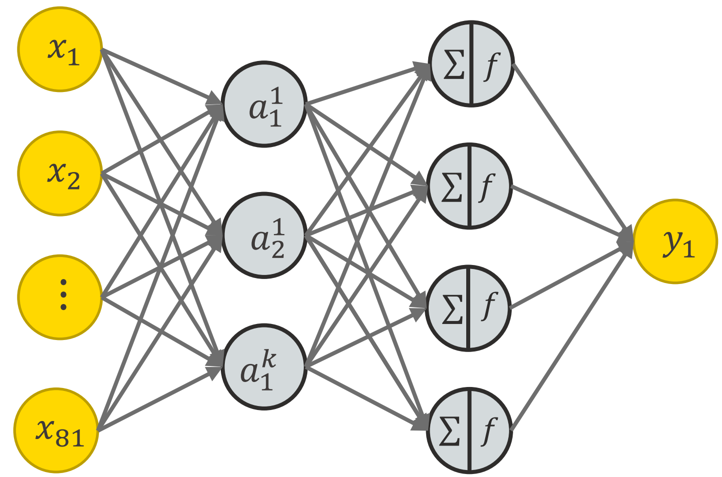
- Problem: A lot of variables / weights
  - Example: Image with 224 x 224 pixels with 3 channels and 100 neurons in the next layer ➔ 150,528 inputs ➔ 15,052,800 weights in the first layer.
  - Unmanageable and likely leads to overfitting during training

Problems with FFNN for Image Classification

- Goal: Train network to recognize x’s
- Approach: Flatten the image and apply a feed forward neural network

-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	-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	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

$$\begin{pmatrix} -1 \\ -1 \\ \vdots \\ 1 \\ -1 \\ \vdots \\ -1 \\ -1 \end{pmatrix}$$



- Problem: Loss of spatial dependencies

Challenge: Different Variations

- Goal: Train network to recognize x’s

-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	-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	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

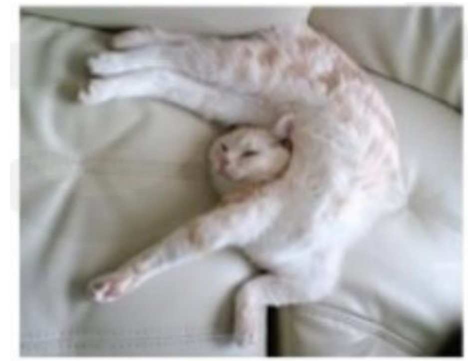
$\overset{?}{=}$

-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	-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	-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

Different Variations

Viewpoint variations

Deformations



Illumination conditions

Intra-class variations

Source: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L3.pdf

Use Filter 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

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

Convolution & pooling layers

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

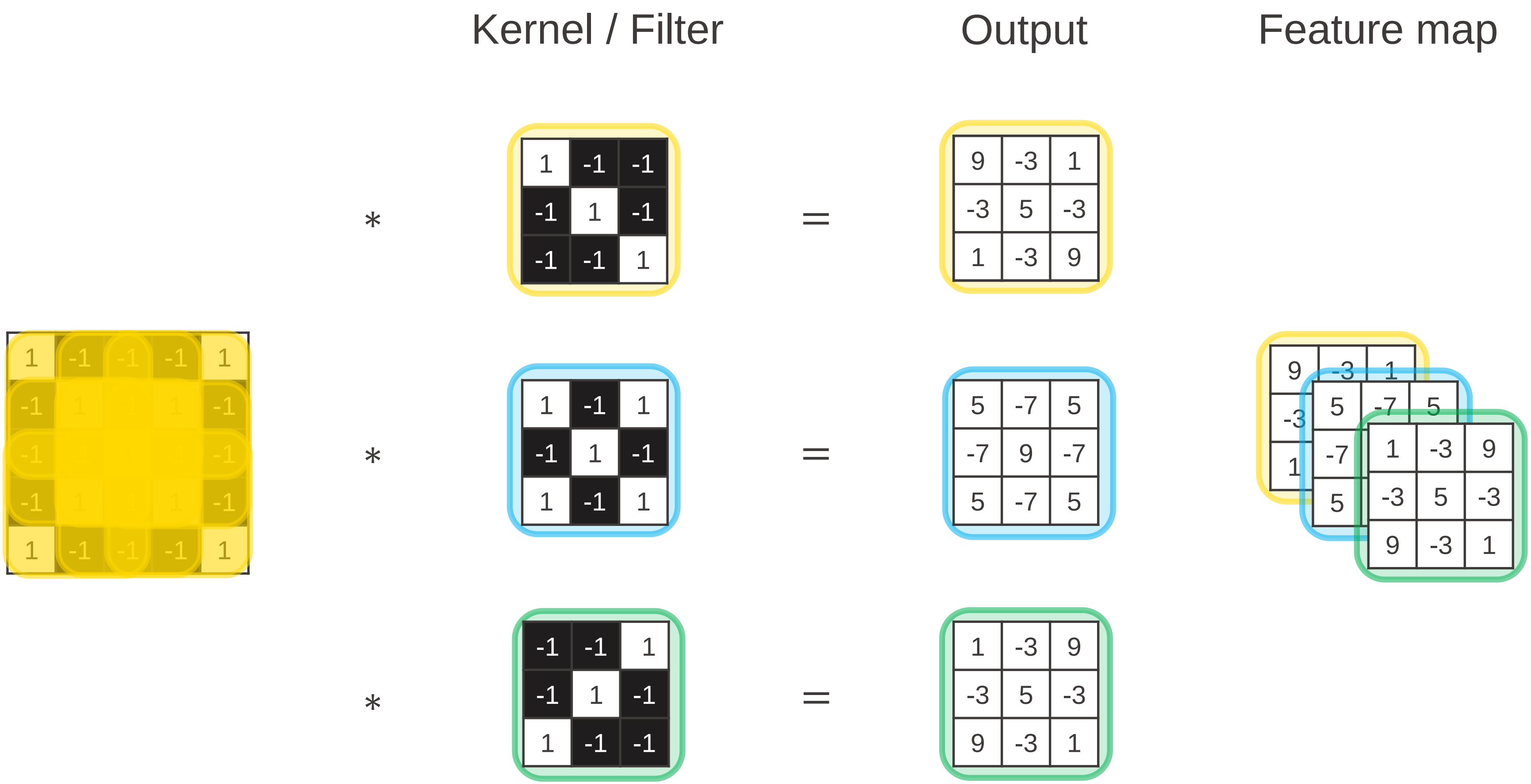
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$$* \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} = \begin{array}{l} \mathbf{1} * \mathbf{1} + (-\mathbf{1}) * (-\mathbf{1}) + (-\mathbf{1}) * (-\mathbf{1}) \\ +(-\mathbf{1}) * (-\mathbf{1}) + \mathbf{1} * \mathbf{1} + (-\mathbf{1}) * (-\mathbf{1}) \\ +(-\mathbf{1}) * (-\mathbf{1}) + (-\mathbf{1}) * (-\mathbf{1}) + \mathbf{1} * \mathbf{1} = \mathbf{9} \end{array}$$

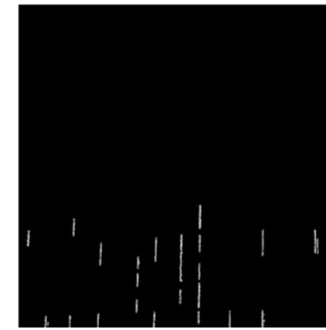
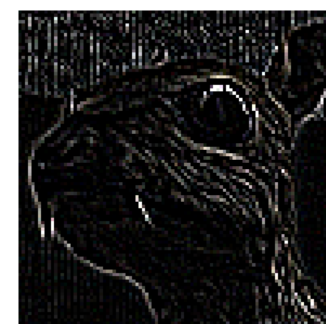
$$* \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} = \begin{array}{l} (-\mathbf{1}) * \mathbf{1} + (-\mathbf{1}) * (-\mathbf{1}) + \mathbf{1} * (-\mathbf{1}) \\ +(-\mathbf{1}) * (-\mathbf{1}) + \mathbf{1} * \mathbf{1} + (-\mathbf{1}) * (-\mathbf{1}) \\ +\mathbf{1} * (-\mathbf{1}) + (-\mathbf{1}) * (-\mathbf{1}) + (-\mathbf{1}) * \mathbf{1} = \mathbf{1} \end{array}$$

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.

Applying Multiple Filters



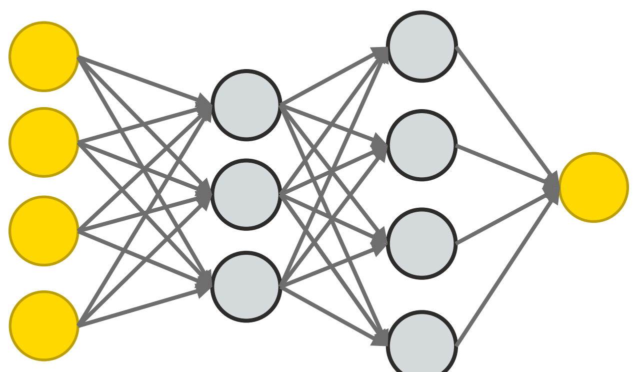
Impact of Handcrafted Kernel

	Original image	Kernel	Result
Vertical line detection		$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	
Edge detection		$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

One Way to Classify Images

1. Use domain knowledge to define important features
2. Try to detect these features

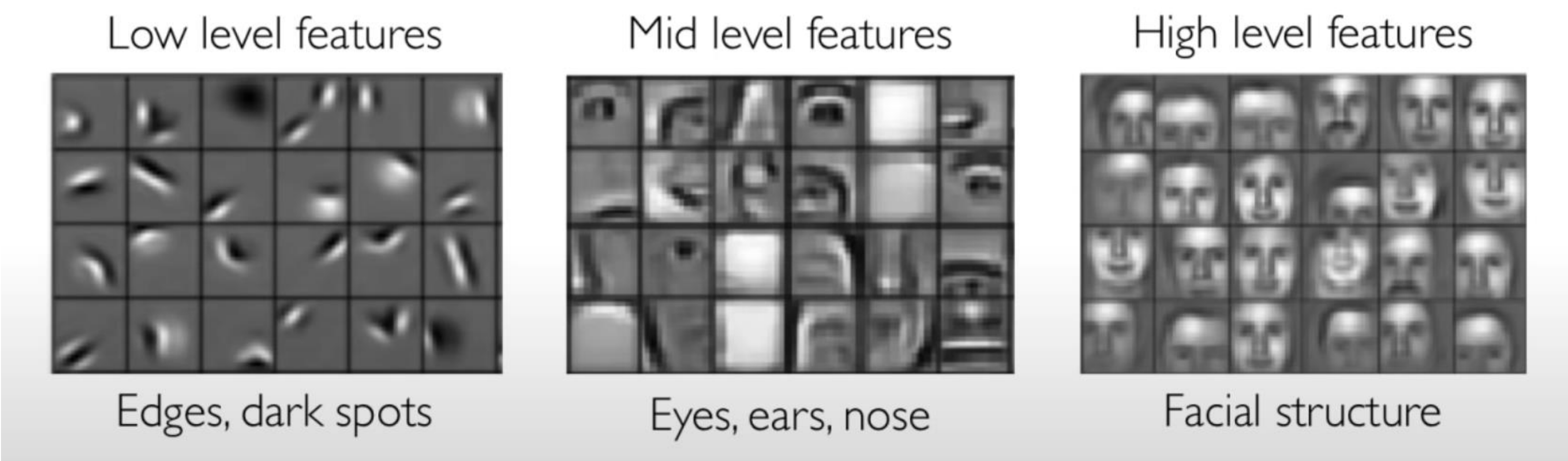
- Problem: Handcrafting different filters is hard
- Solution: Use a Convolutional Neural Network (CNN)
  - Kernel / filters trained as part of the network to extract features
  - Extracted features are used by the network for the classification task



Cat

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.

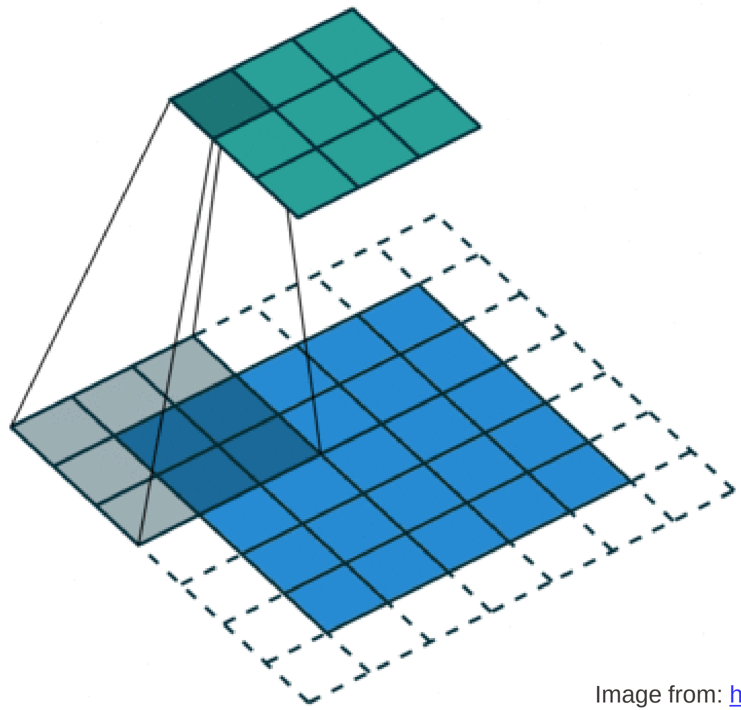
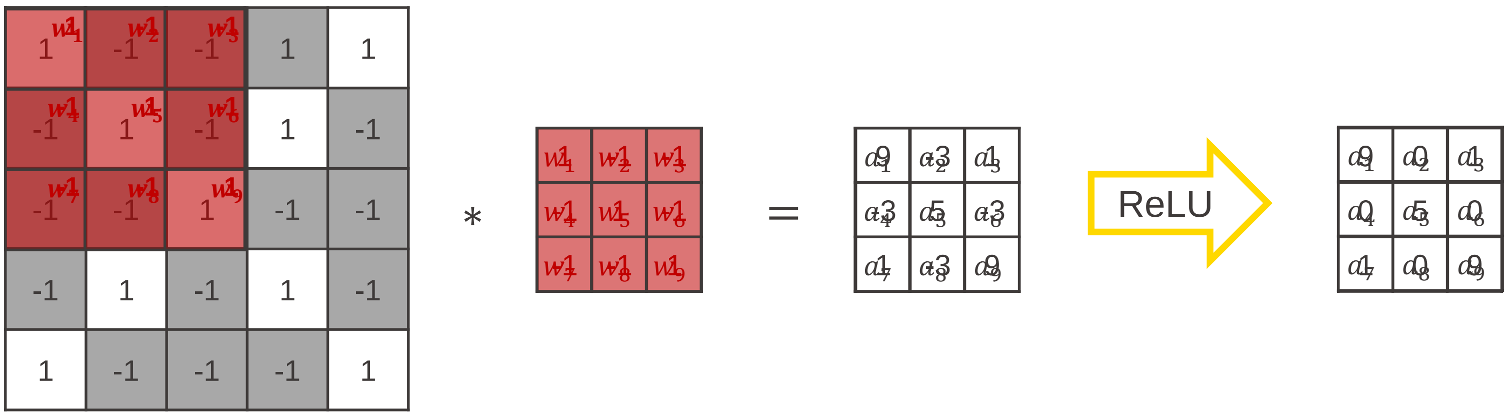


Image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

How Do Convolutional Layers Work?

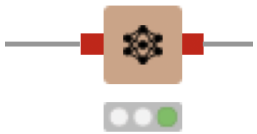
- Idea:
  - Use a kernel / weight matrix and slide it over the image
  - At each position: Apply the convolution and a non-linear activation, e.g. ReLU

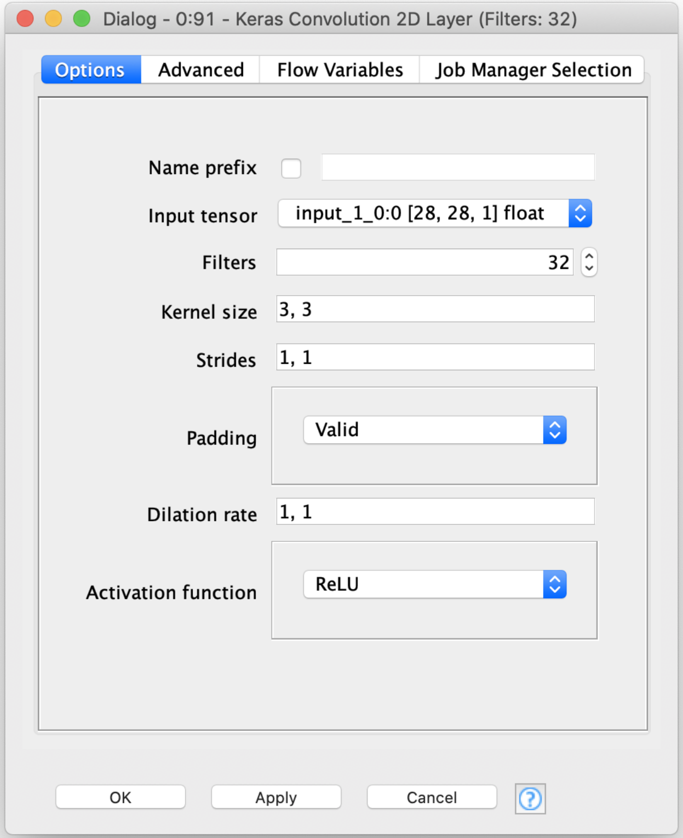


- The weights of the kernel are learned during training
- Note: These are similar steps like in a feed forward neural network
  - Convolution ≙ Weighted sum of inputs

Keras Convolution 2D Layer

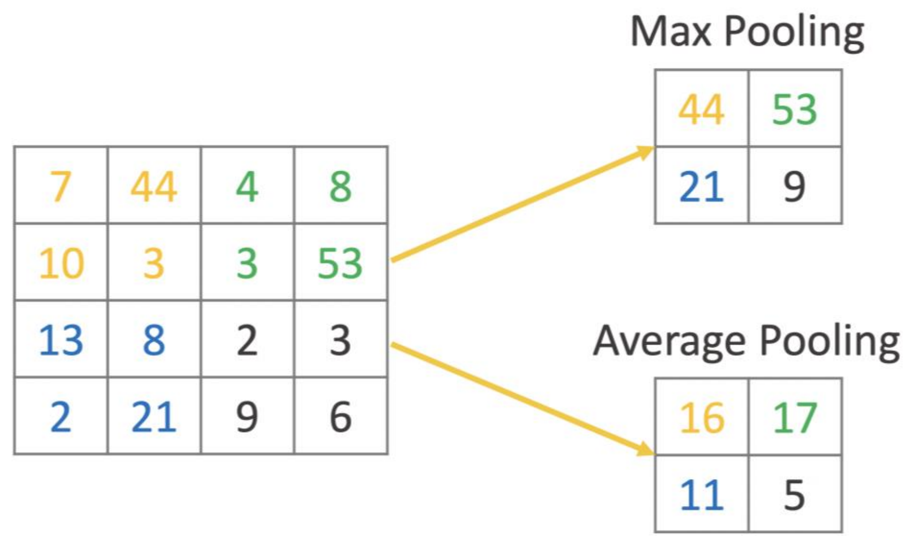
Keras Convolution
2D Layer



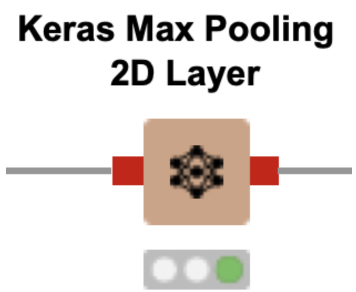


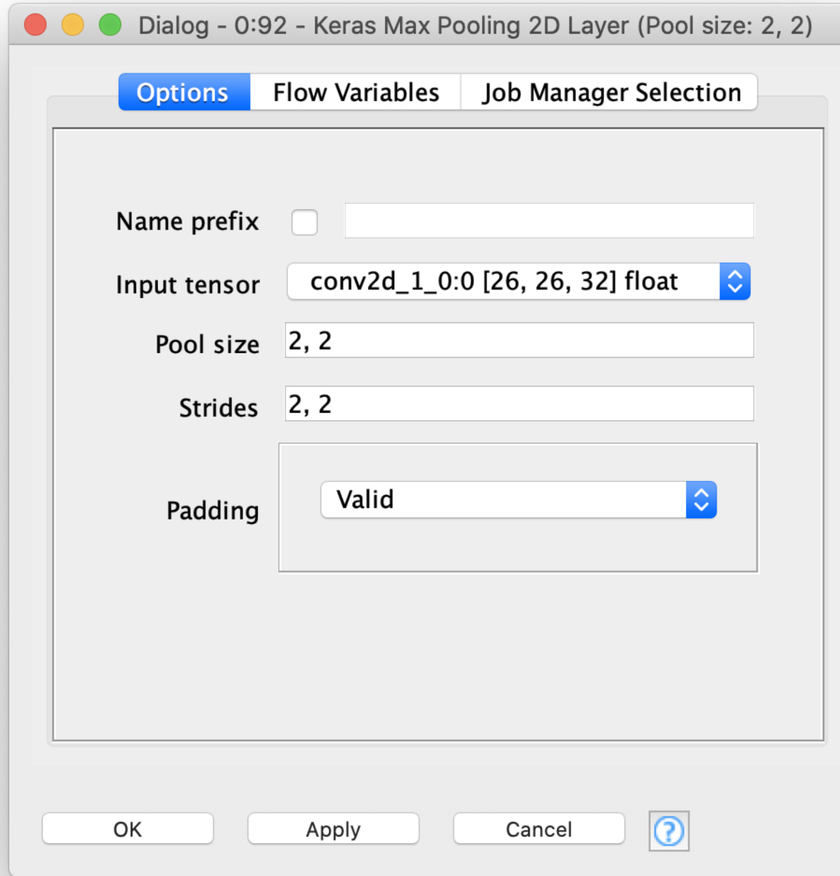
Pooling Layer

- 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 in between convolutional layers to
  - Increase the receptive field of the following layers
  - Reduce computational complexity
- No parameters to learn



Keras Max Pooling 2D Layer Node





CNN for Image Classification

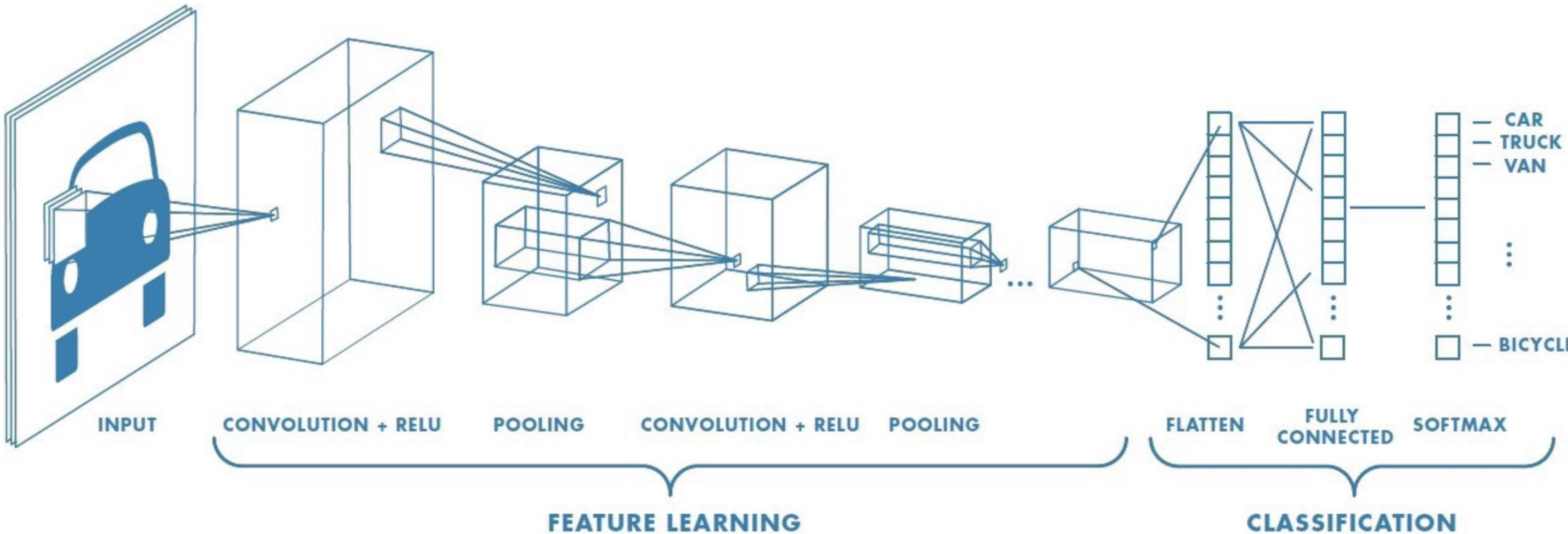


Image from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Image Classification: Cats & Dogs Data

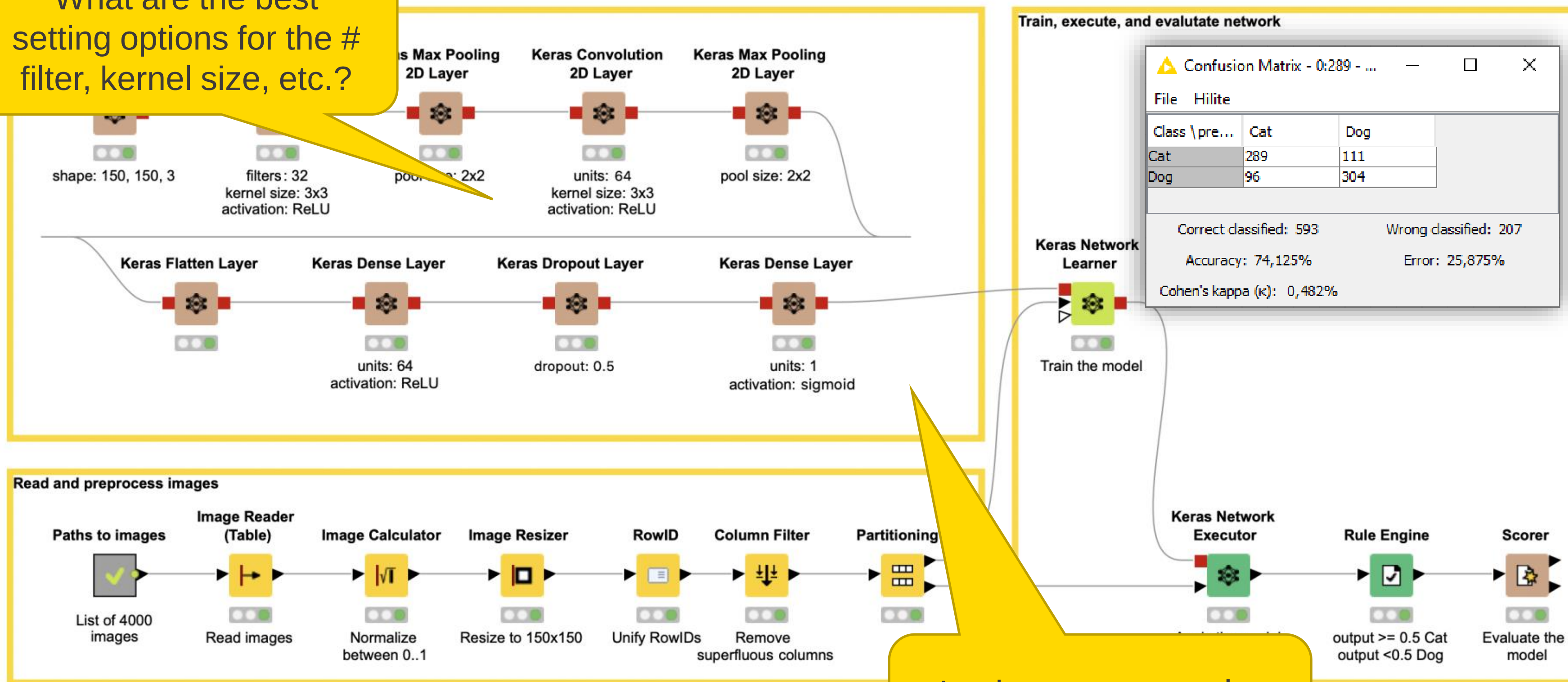
Kaggle Dogs vs Cats Challenge
https://www.kaggle.com/c/dogs-vs-cats/overview

Classification with KNIME

Image	Class	Prediction
	Cat	Cat
	Dog	Dog
	Cat	Cat

Simple CNN for Image Classification

What are the best setting options for the # filter, kernel size, etc.?



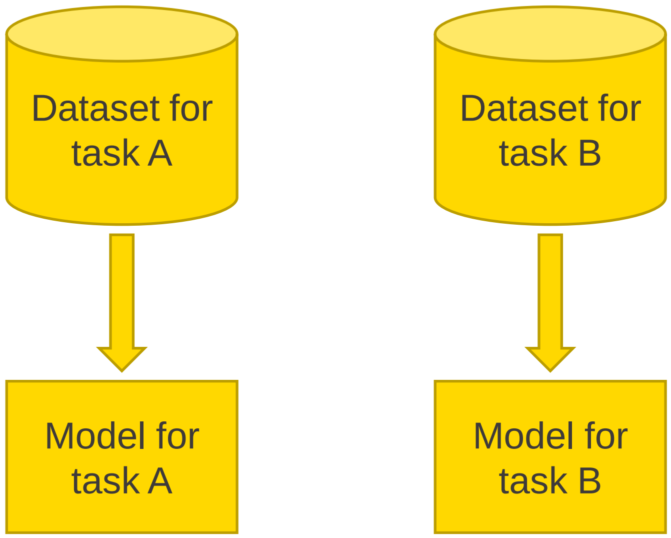
Look at some popular CNNs

Download the workflow from the [KNIME Hub]

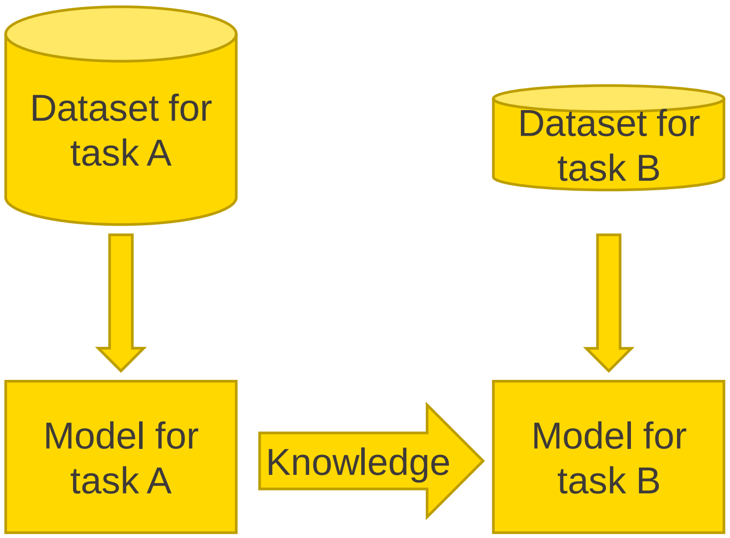
Transfer learning

Standard vs. Transfer Learning

Standard learning



Transfer learning



Cancer Cell Classification with Transfer Learning

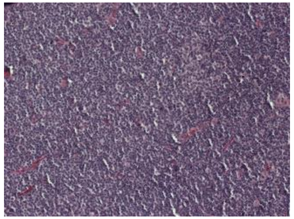
- Transfer learning can be adapted to a wide range of image classification problems
- Task: Classify histopathology slide images and about the type of lymphoma
  - chronic lymphocytic leukemia (CLL)
  - follicular lymphoma (FL)
  - mantle cell lymphoma (MCL)
- Reuse VGG16 network

Original Task

VGG16 → Cat

This Photo by Unknown Author is licensed under CC BY-SA

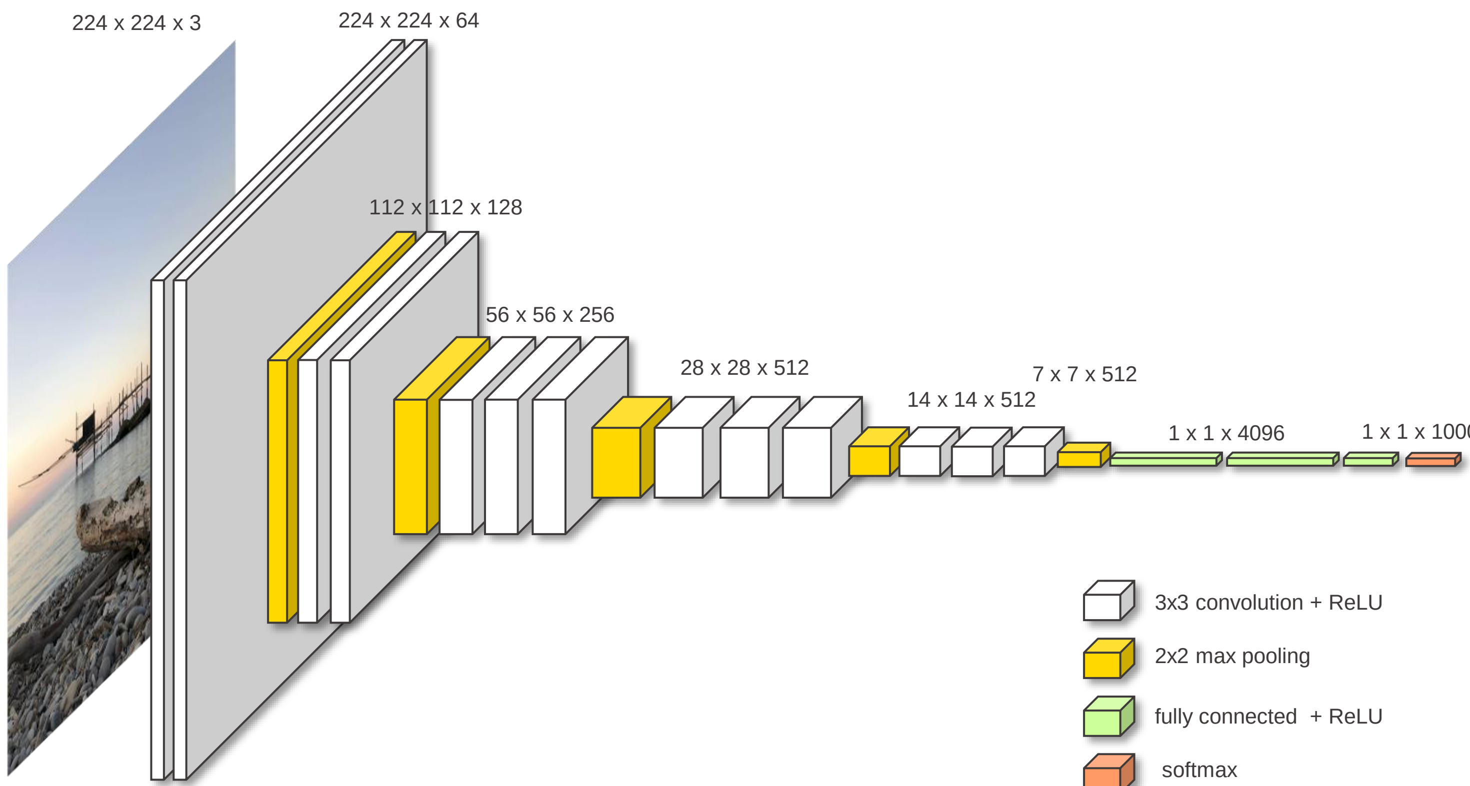
New Task



VGG16 → New → CLL

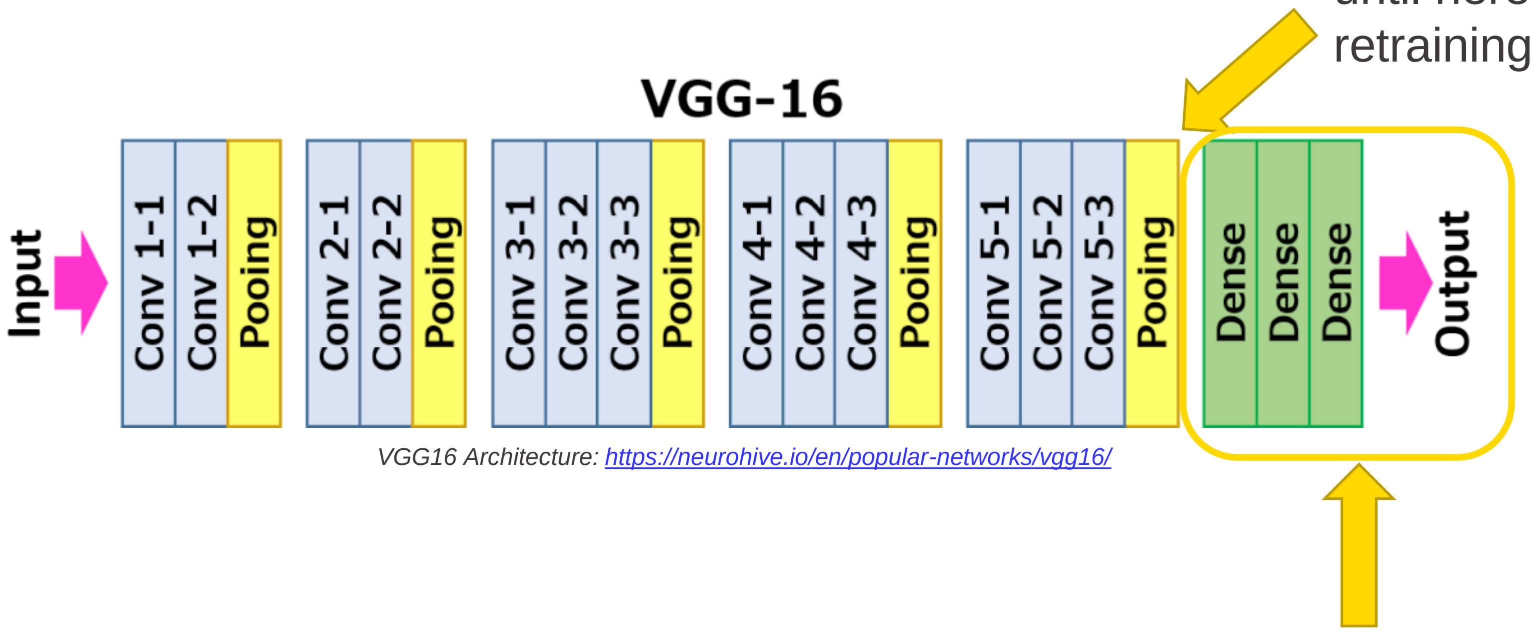
Image From: https://ome.grc.nia.nih.gov/iicbu2008/lymphoma/index.html

Popular CNN: VGG-16 (2015)



Transfer Learning for Image Classification

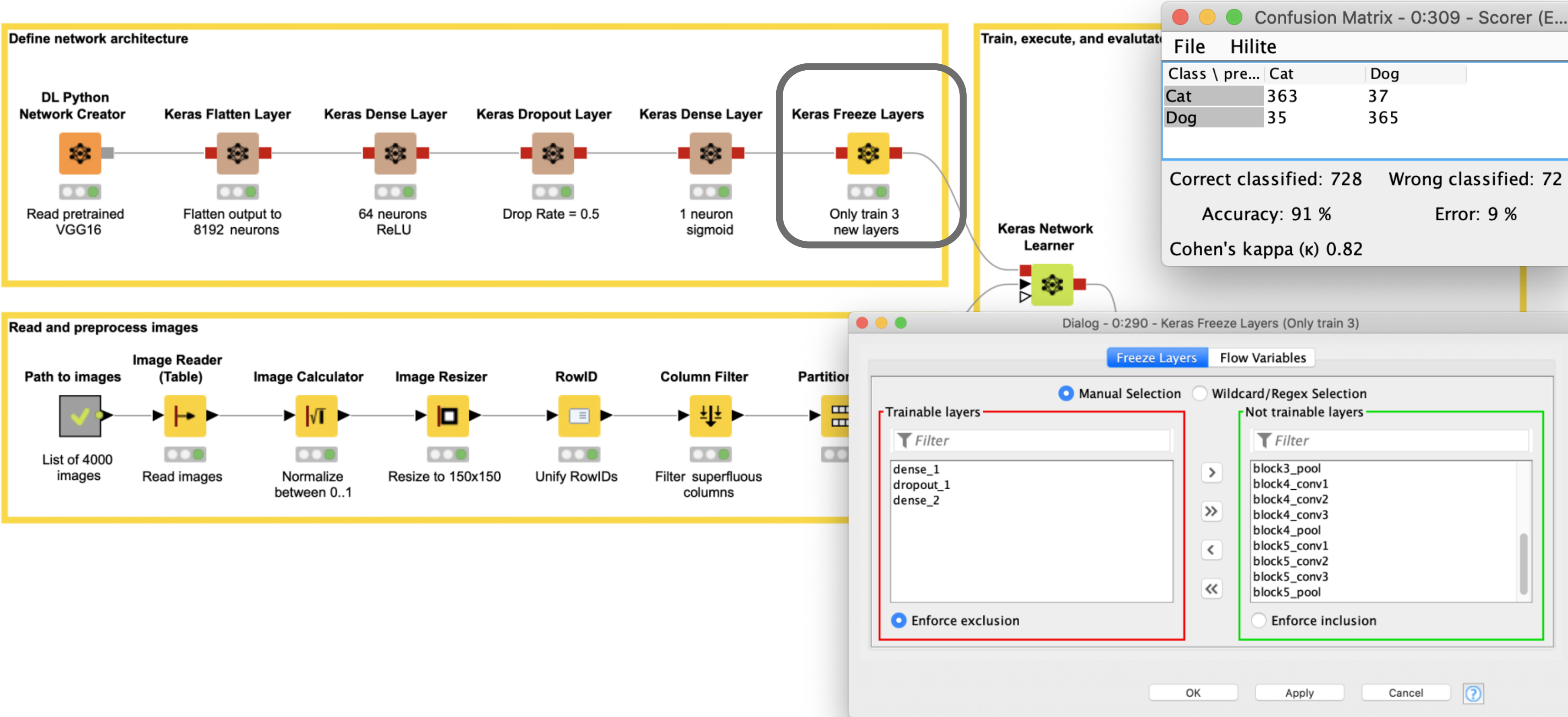
Use all layers until here, without retraining



VGG16 Architecture: https://neurohive.io/en/popular-networks/vgg16/

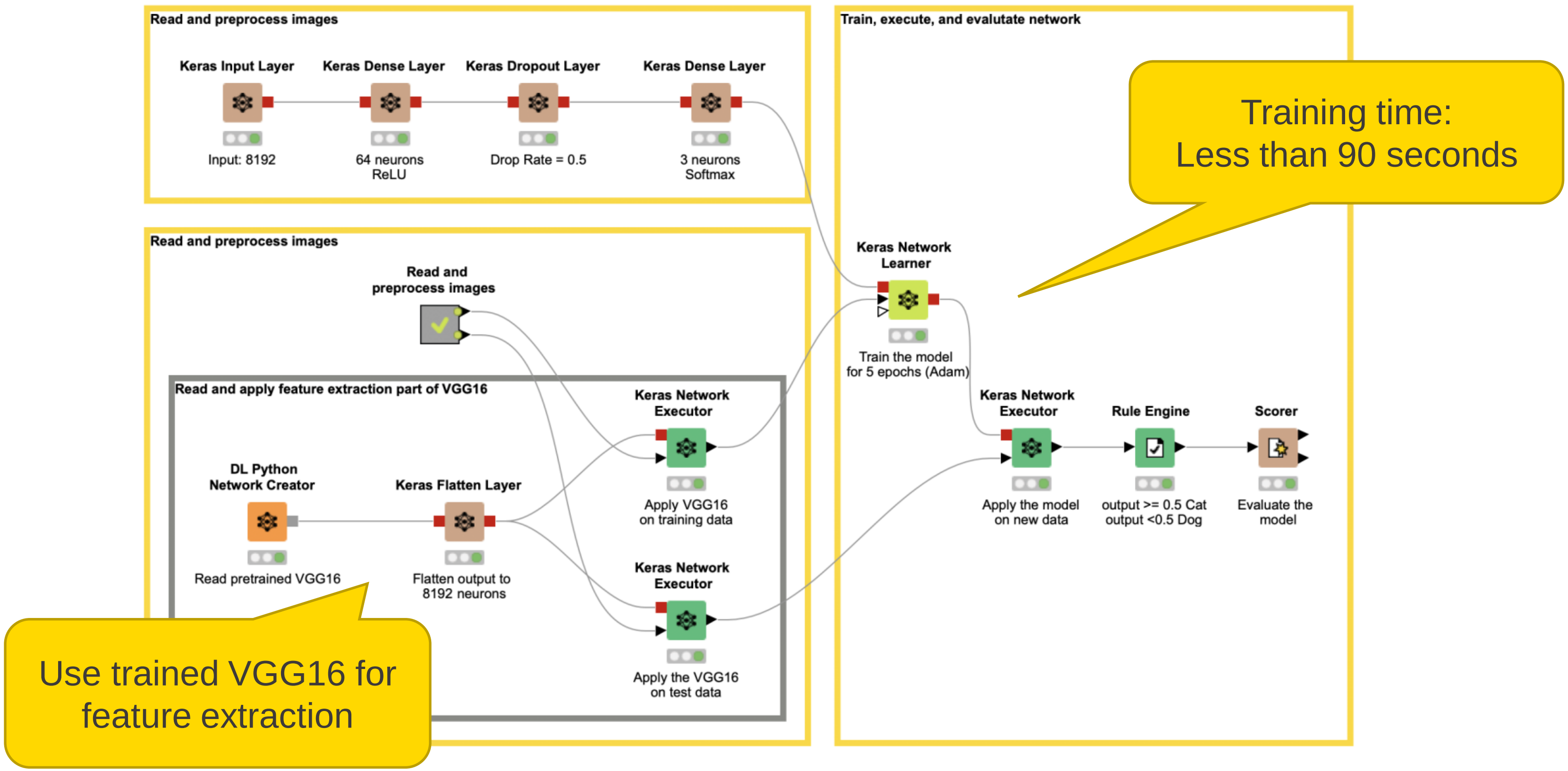
Replace and retrain this part

Transfer Learning for Image Classification



Download the workflow from the [KNIME Hub]

Transfer Learning for Image Classification: Option 2



Download the workflow from the [KNIME Hub]

Sequential data & RNN

Text Generation: What Is The Next Word?

- Text is a sequence of words or characters → sequential data
- Task: Predict the next word in a sequence

- The last word in the sequence is “the”. What is the next word?

tree ?

Example workflows are available on the

elephants ?

hub ?

Requirement 1: The network must be able to take context information into account

Text Generation: What Is The Next Word?

- The hotel was good, not bad at all.
  This made our vacation _______
- The hotel was bad, not good at all.
  This made our vacation _______

Requirement 2: The network must be able to take order into account

Text Generation: What Is The Next Word?

Example 1:

Hi Suzie,

Input length = 2 words

Example 2:

Hi Suzie,
Our week in the alps
was amazing.
Looking forward to

Input length = 12 words

Requirement 3: The network needs to handle different sequence lengths

Text Generation: What Is The Next Word?

- Time information can be either in the beginning, the middle, or the end of a sentence

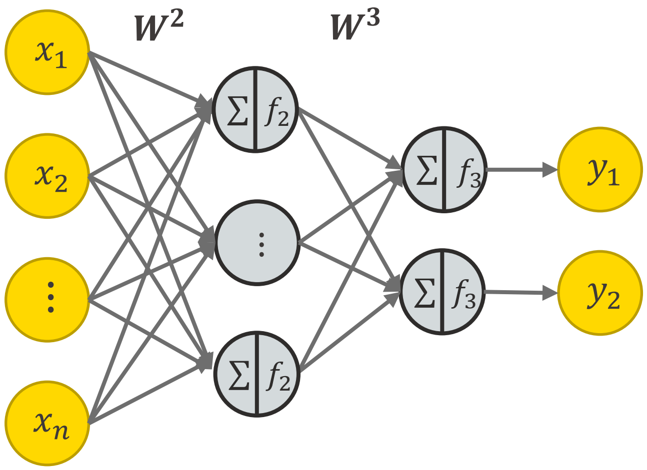
Yesterday, I was shopping. I will wear my new …

I was shopping, yesterday. I will wear my new …

Requirement 4: The network must be position independent

Feed Forward Neural Network for Sequential Data

- Idea: Use a feed forward neural network to handle sequential data
- Doesn't meet the requirements of sequential data
  - Doesn't take word order into account
  - Fixed input ➔ can't handle different sequence length
  - Doesn't share parameters ➔ not position invariant
- Solution: Recurrent Neural Network (RNN)

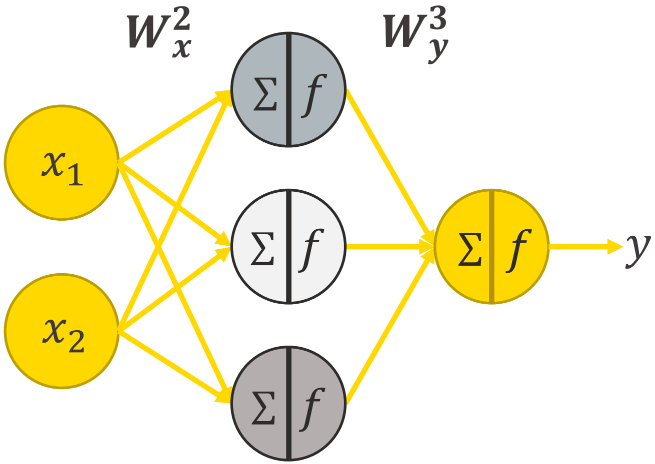


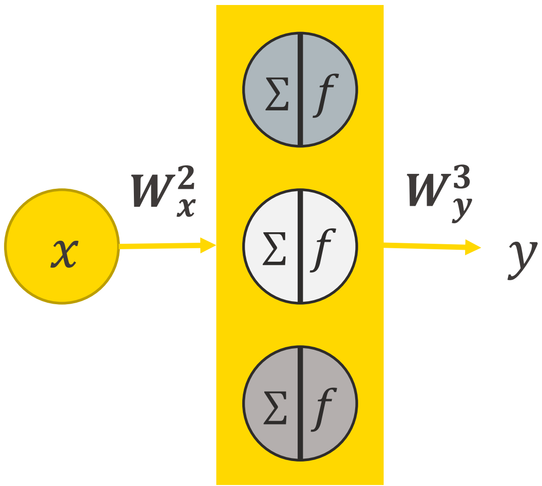
Recurrent Neural Networks

- **R**ecurrent **N**eural **N**etworks (RNNs) are a family of neural networks suitable for processing of sequential data
- Key idea: use a loop connection

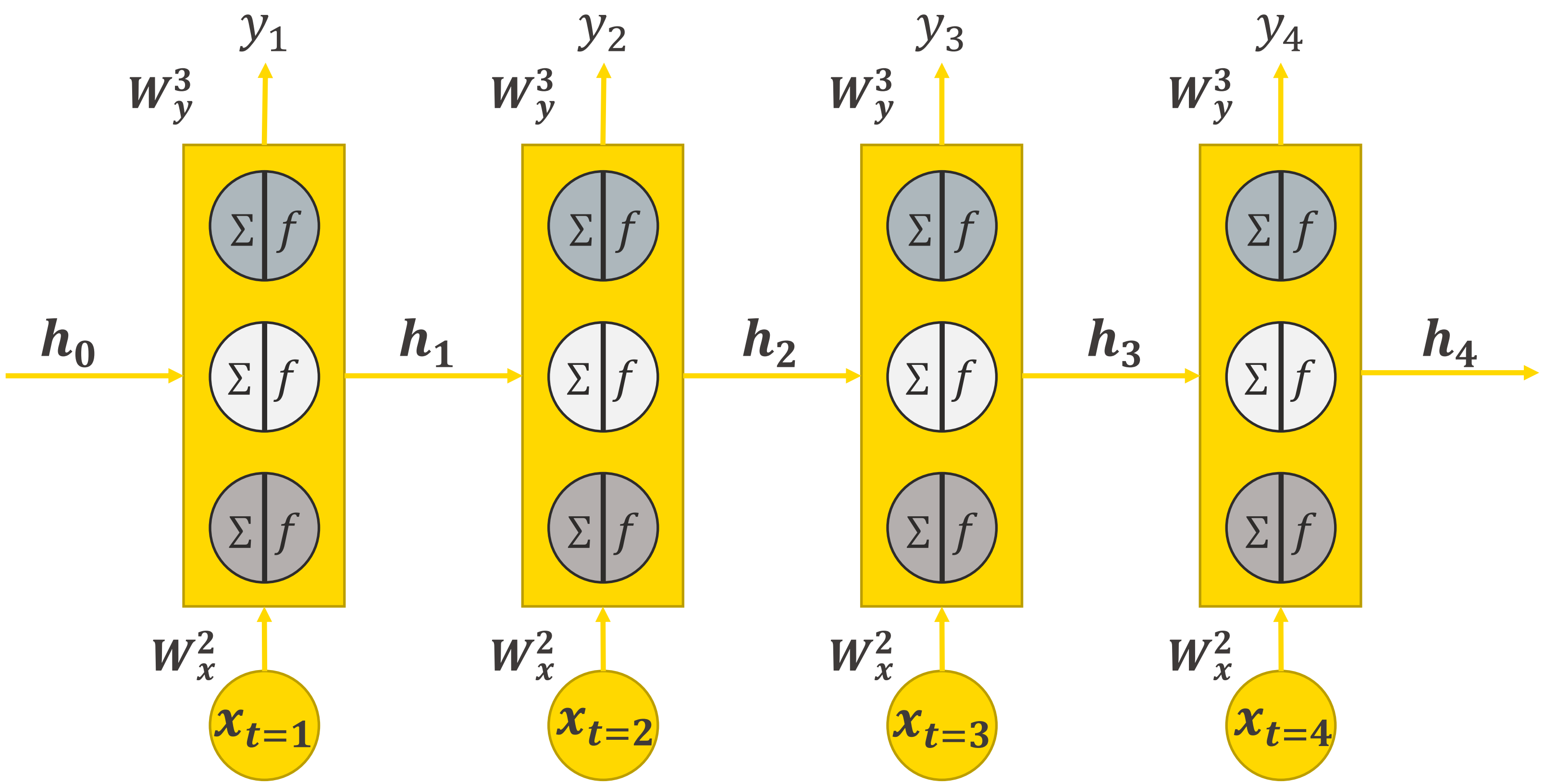
- RNNs are used for all sorts of tasks:
  - Language modeling / Text generation
  - Text classification
  - Neural machine translation
  - Text summarization
  - Image captioning
  - Speech to text
  - Demand prediction
  - Stock price prediction
  - …

From Feed Forward to Recurrent Neural Networks

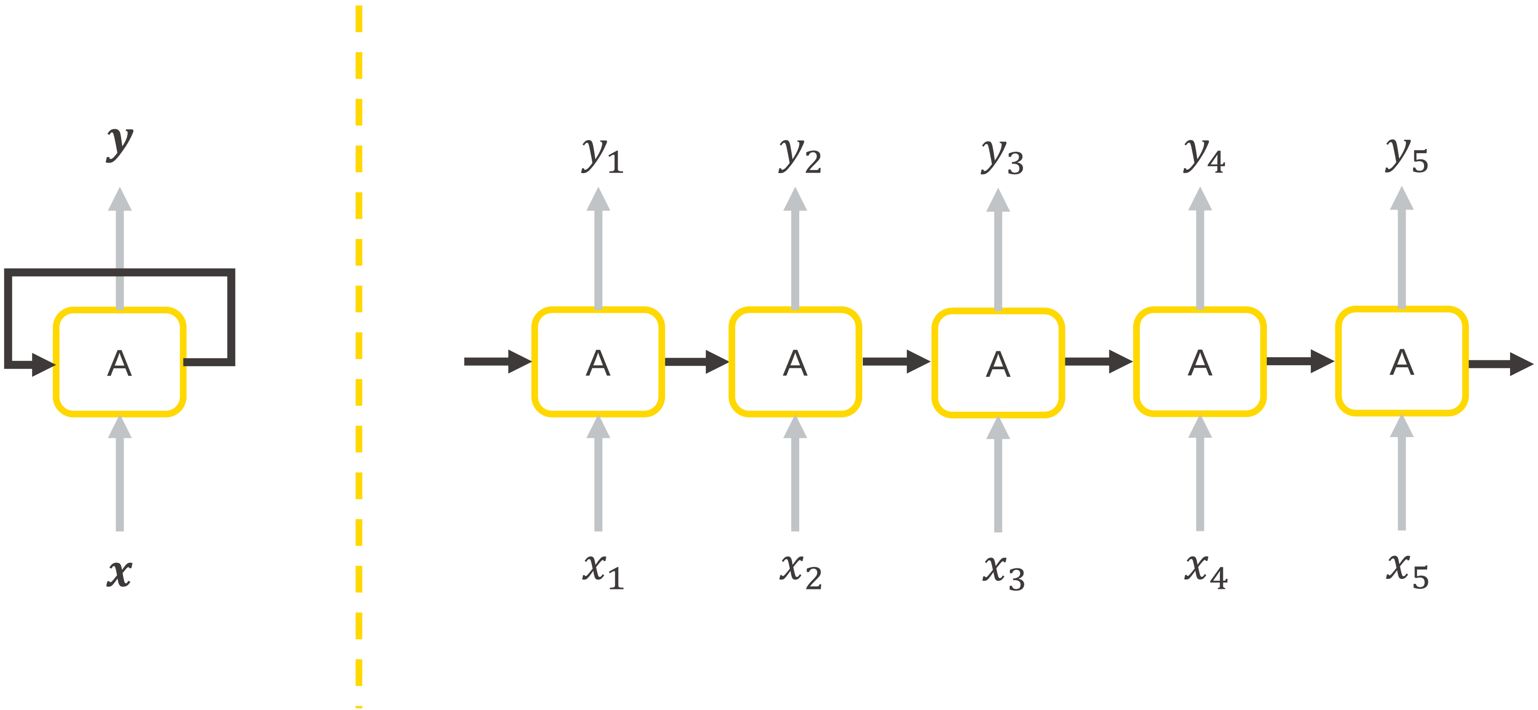


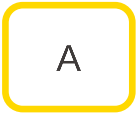


From Feed Forward to Recurrent Neural Networks



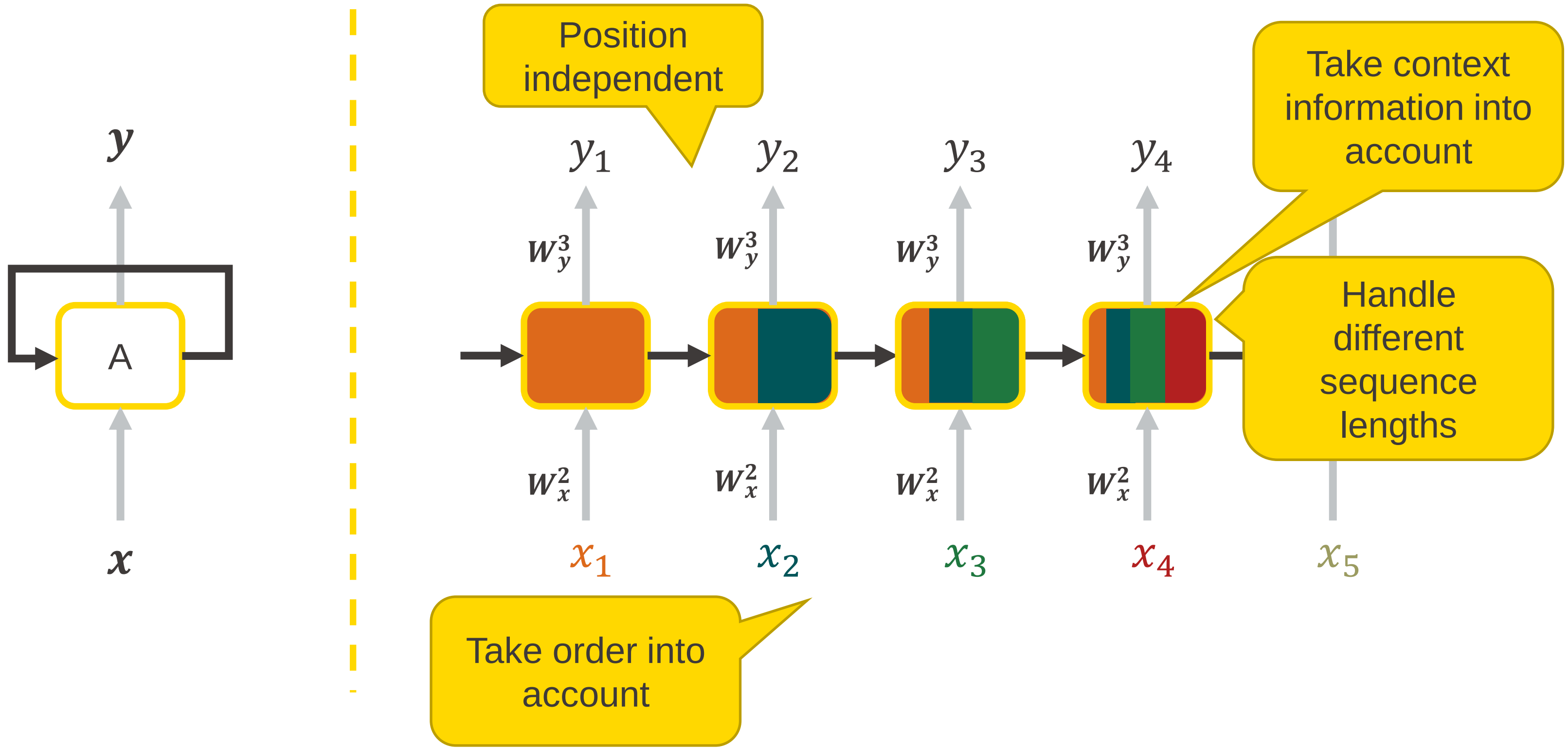
RNN Rolled and Unrolled Representation





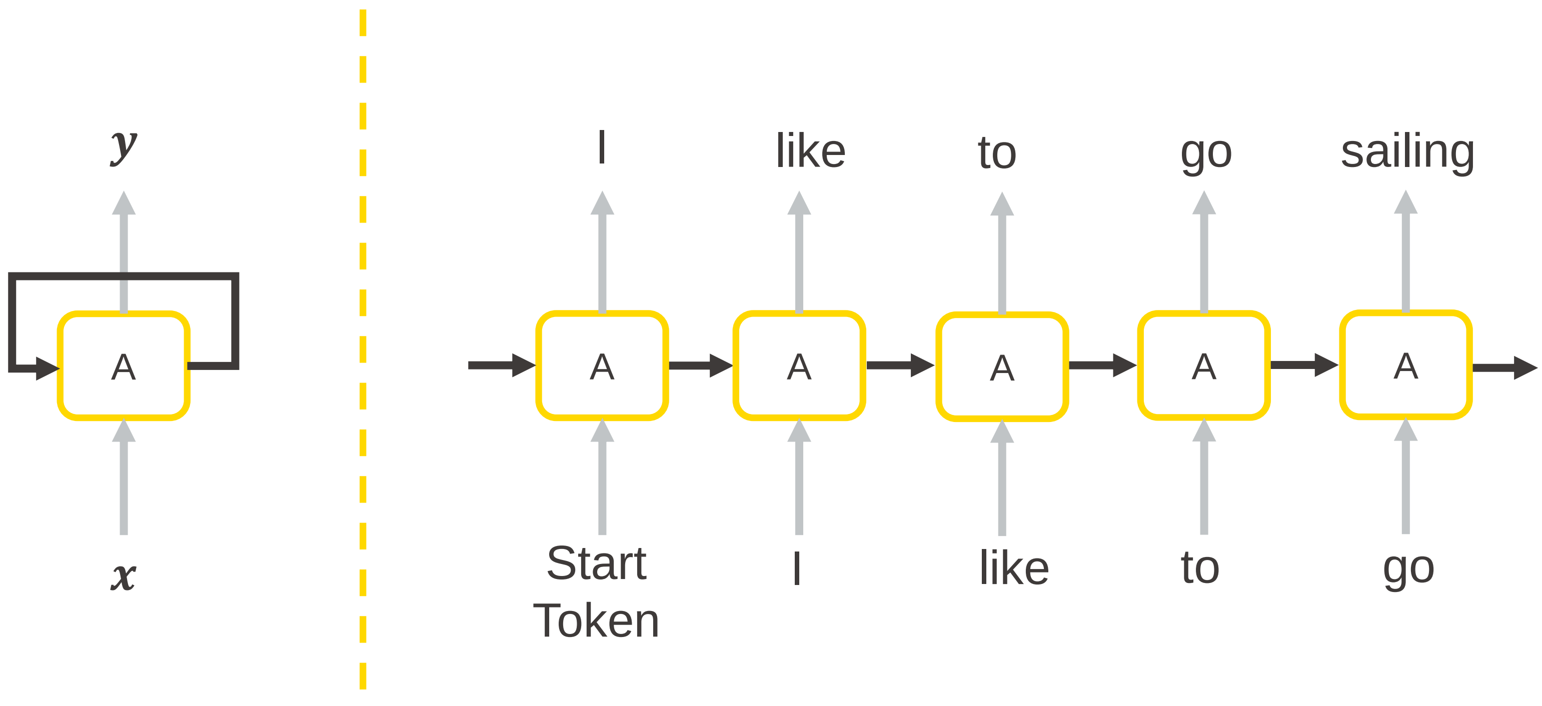
= A feed forward network with one or multiple layers

Memory of an RNN



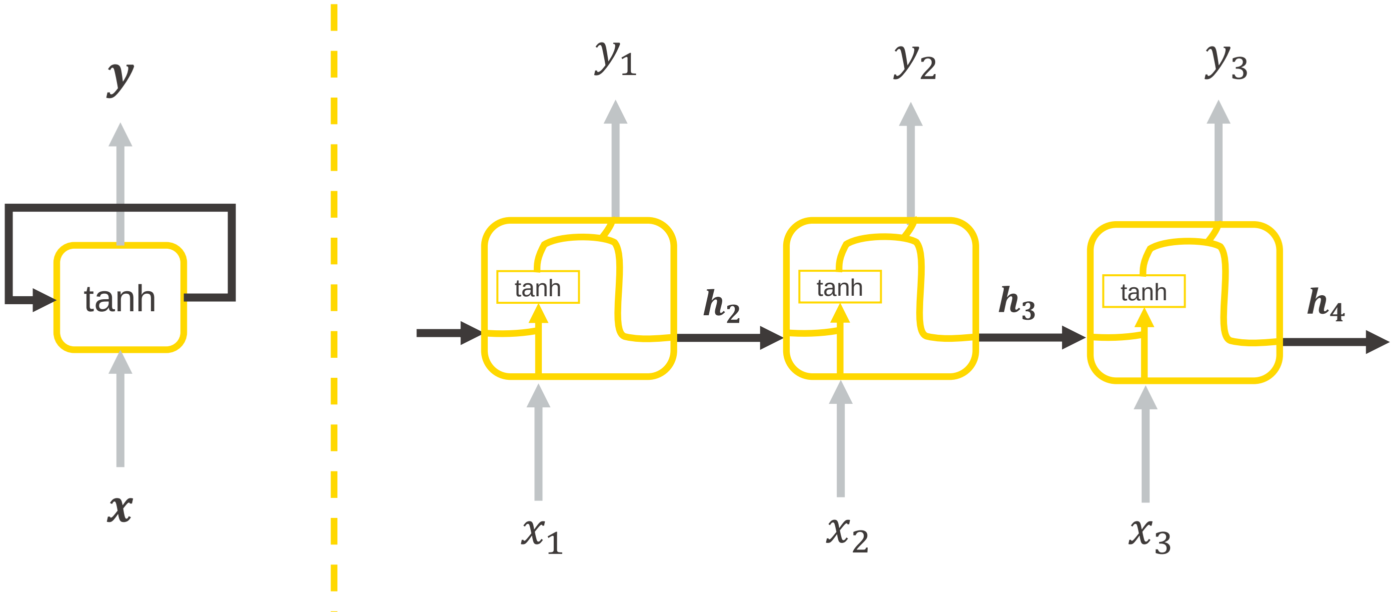
A = A feed forward network with one or multiple layers

Example: Language model



A = A feed forward network with one or multiple layers

The Math Behind An RNN



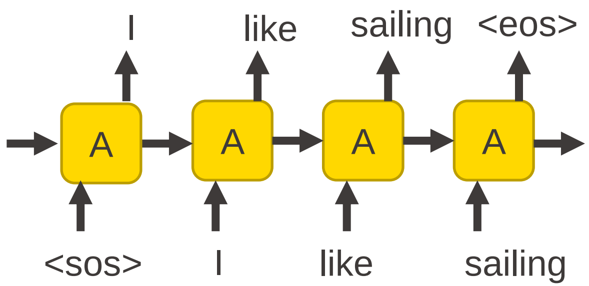
$$y_1 = \tanh(W_x x_1 + W_h h_0)$$

$$y_2 = \tanh(W_x x_2 + W_h h_1)$$

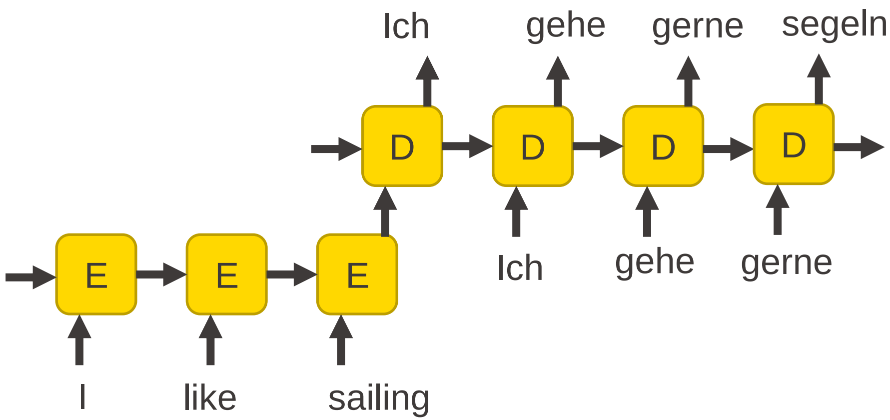
$$\vdots$$

$$y_n = \tanh(W_x x_n + W_h h_{n-1})$$

RNN Architectures: Many to Many (Seq2Seq)



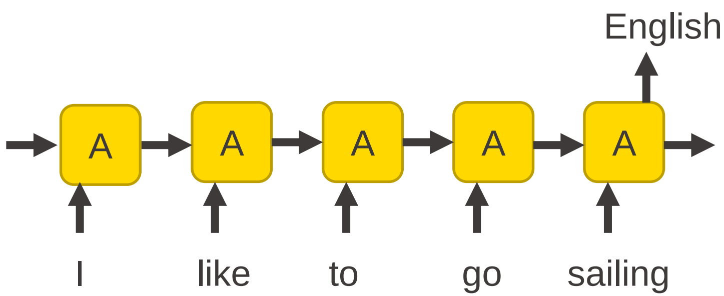
Language model



Neural machine translation

RNN Architectures: Many-to-One & One-to-Many

Many-to-one



Language classification

One-to-many

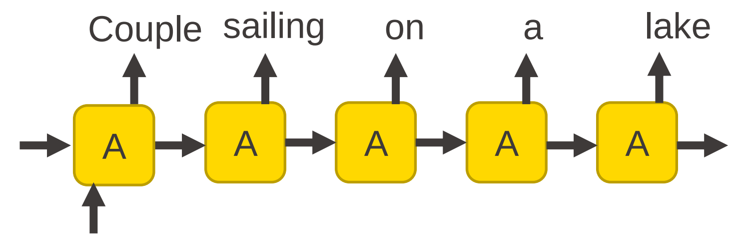
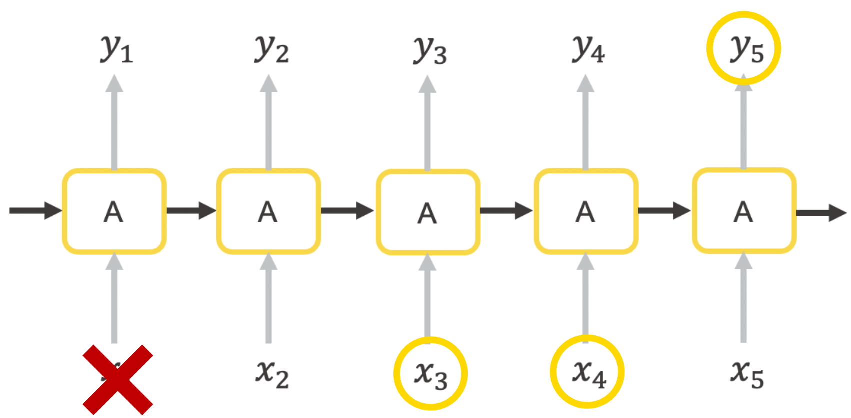


Image captioning

Limitation of Simple RNNs

The “memory” of simple RNNs is sometimes too limited to be useful:

- *“Cars drive on the ____” (road)*
- *“I love the beach.*
  My favorite sound is the crashing of ____“ (cars? glass? waves?)



Long short-term memory (LSTM)

Long Short-Term Memory (LSTM) Units

Special type of unit with

- an additional cell state
- three gates
  - Forget gate
  - Input gate
  - Output gate

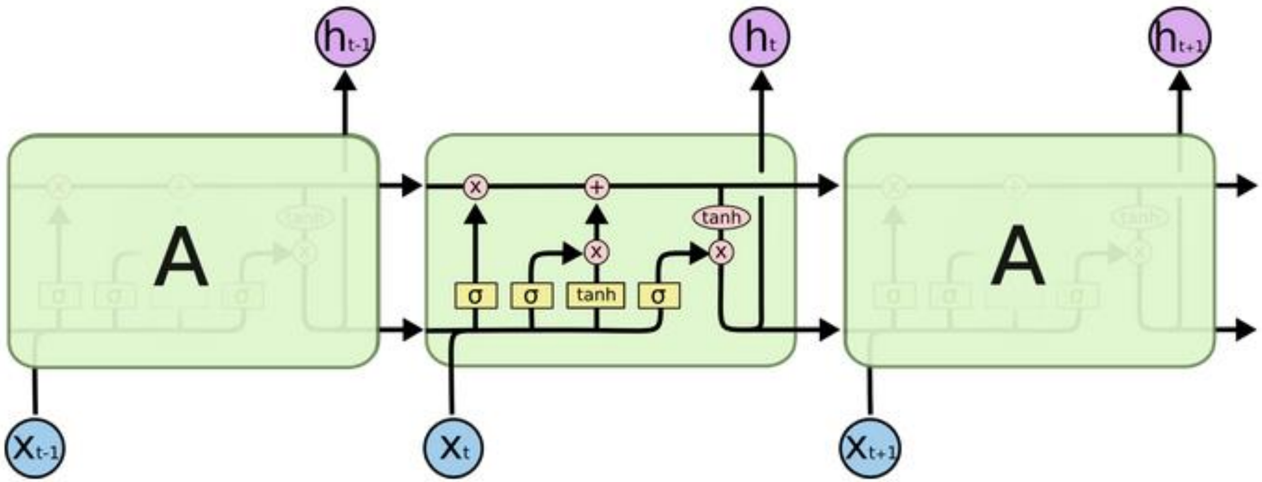


Image Source: Christopher Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Idea of a Gate

- A gate can be …

	open	1	Let all information through
	partially open	(0,1)	Let part of the information through
	closed	0	Let no information through

LSTM Units

- Additional cell state makes it easier to remember information
- At each time step
  1. The forget gate removes irrelevant information from the cell state
  2. The input gate decides which information should be added to the cell state
  3. The cell state is updated
  4. The output gate decides which information to output and to send to the next time step

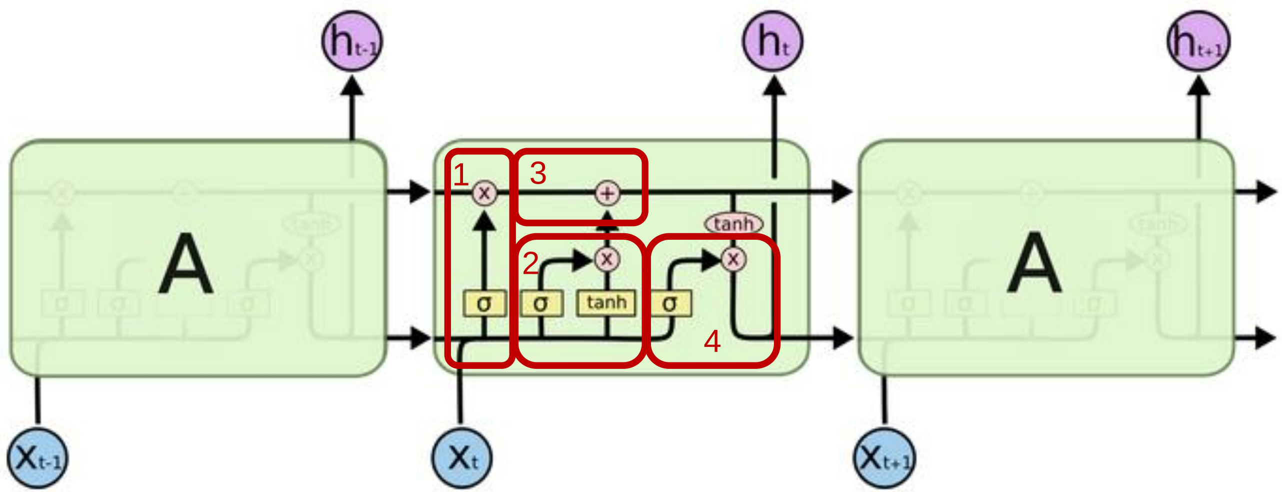
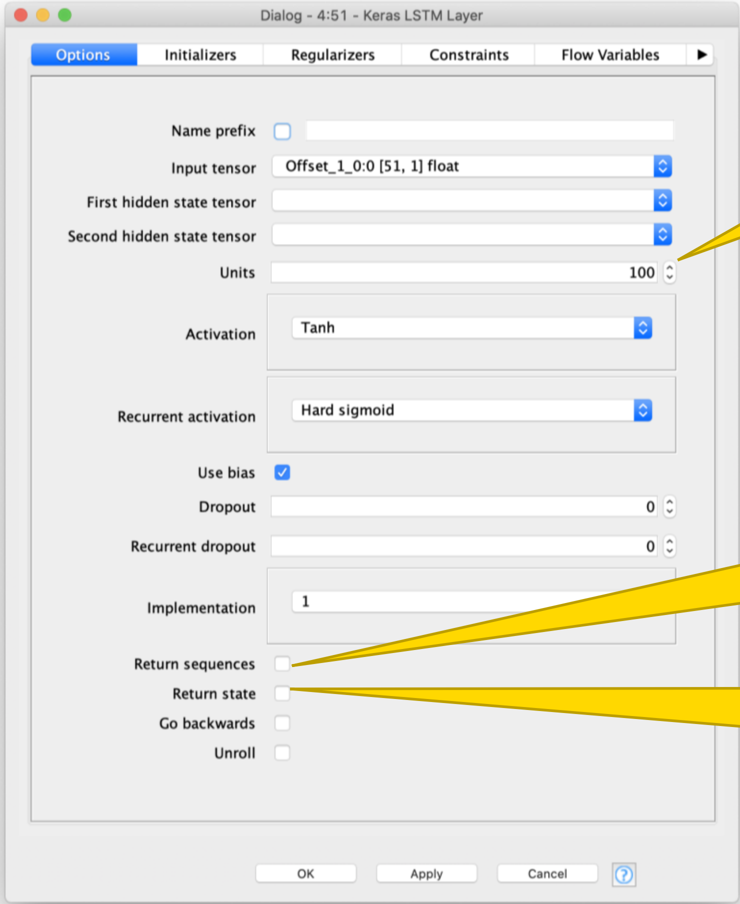


Image Source: Christopher Olah, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Layer Node

Keras LSTM Layer



Size of the hidden state vectors

Activate "Return sequences" for seq2seq models

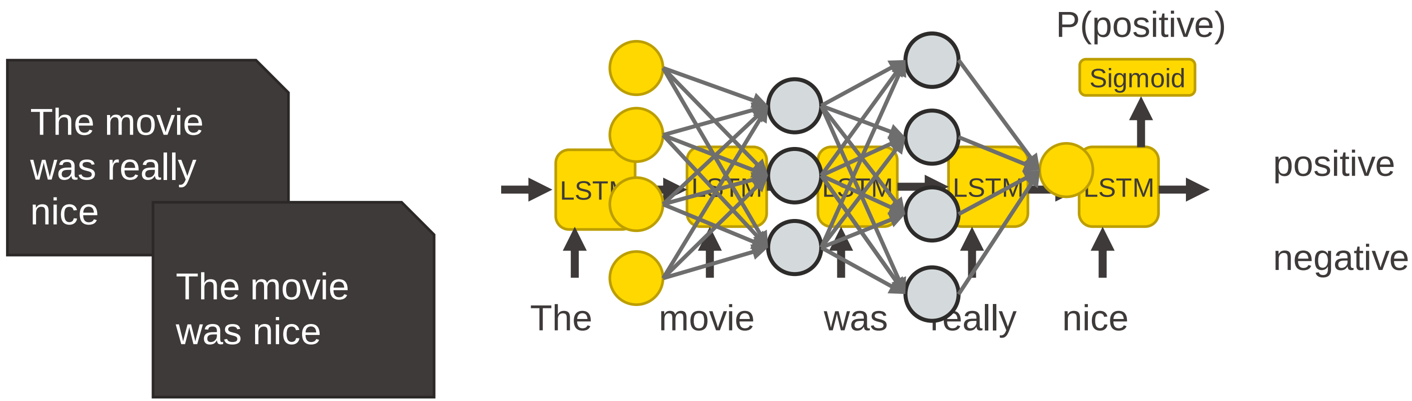
Activate "Return state" to use it during deployment

Optional input ports for the hidden state tensors

Example: Text Classification

- Task: Assigning tags or categories to text according to its content
- Examples:
  - Identify the underlying sentiment of movie / restaurant / product reviews, tweets etc.
    - Positive
    - Negative
  - Classify vacation reviews. What is the review about?
    - Hotel
    - Flight
    - Booking process

Use Case: Sentiment Analysis of Movie Reviews



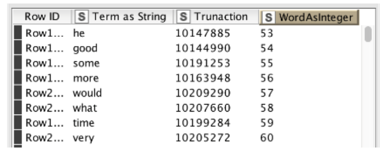
Preprocessing:

- Different sequence lengths
  - Sequences within the same training batch must have the same length
  - Solution: Truncate too long sequences and zero pad too short sequences
- Encoding
  - Index encoding plus embedding layer
  - Large number of different words: Define a fixed dictionary size and assign default (“unknown”) value to all other words

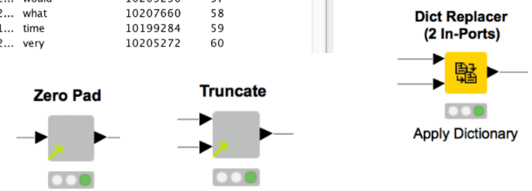
Use Case: Sentiment Analysis of Movie Reviews

Preprocessing

This film is mediocre at best. Angie Harmon is as funny as a bag of hammers. Her bitchy demeanor from Law and Order carries over in a failed attempt at comedy. Charlie Sheen is the only one to come out unscathed in this horrible anti-comedy. The only positive thing to come out of this mess is Charlie and Denise's marriage. Hopefully that effort produces better results.

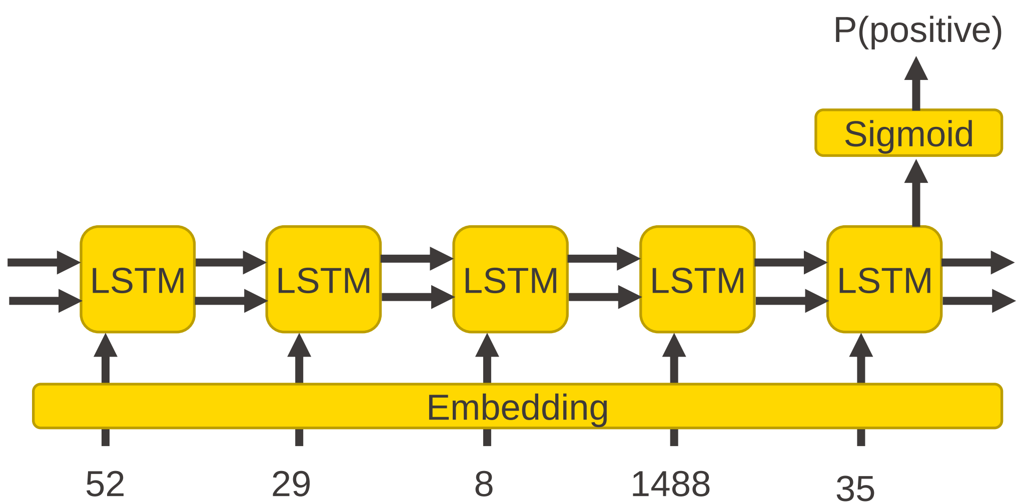


Row ID	Term as String	Truncation	WordAsInteger
Row1...	he	10147885	53
Row1...	good	10144990	54
Row1...	some	10191253	55
Row1...	more	10163948	56
Row2...	would	10209290	57
Row2...	what	10207660	58
Row1...	time	10199284	59
Row2...	very	10205272	60

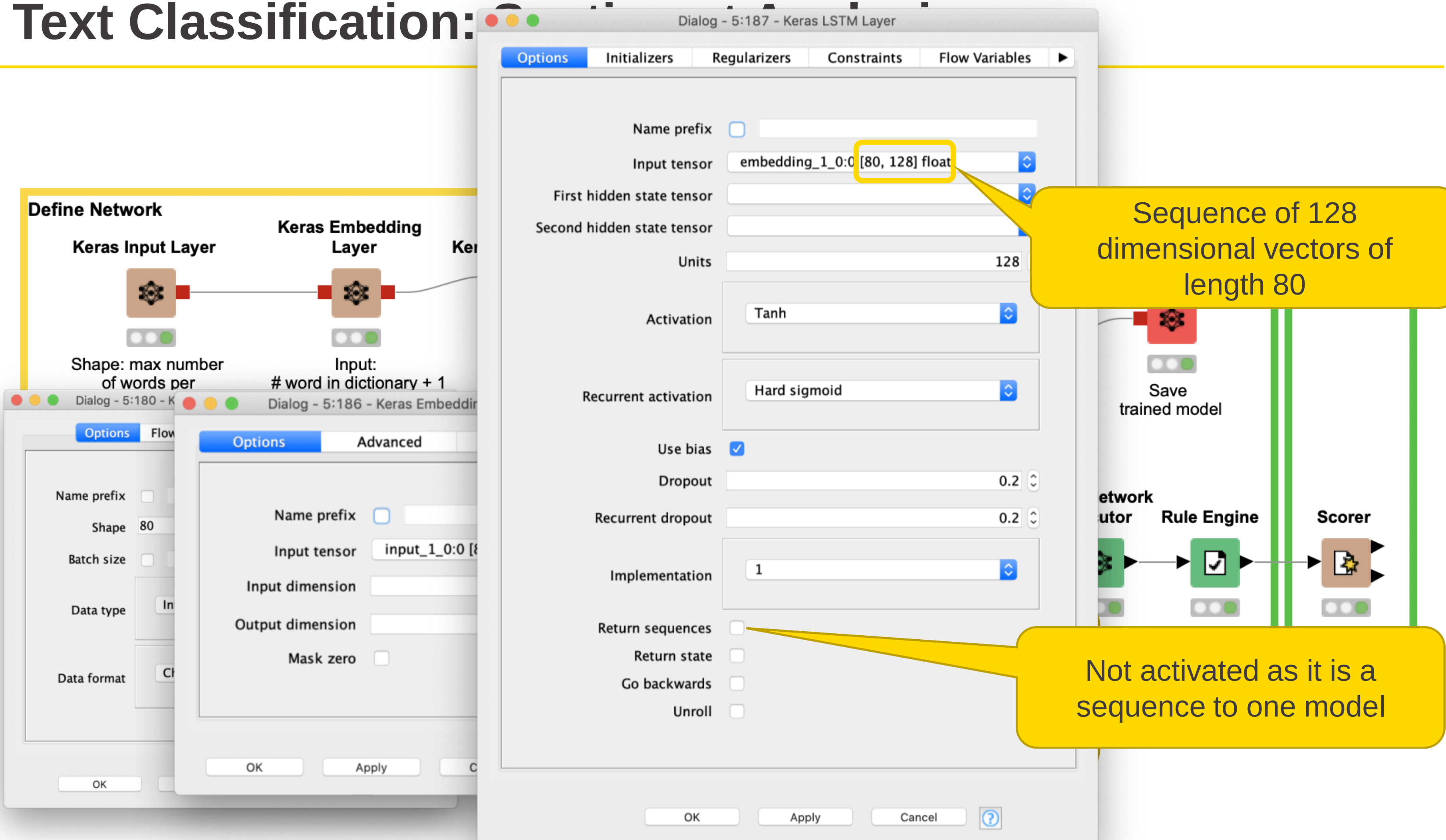


52 29 8 1488 35 130 1 11794 18869 8 20
187 20 3 2984 6 20001 1 883 8167 8409
39 3791 4 8384 2716 148 9 3 1154 585 35
258 1 1987 5781 8 2 65 30 7 221 44
13964 9 10 552 20001 1 18 65 1124 159 7
221 44 6 10 931 8 1987 4 10239 14 1570
1 5136 12 741 7256 131 1829 1

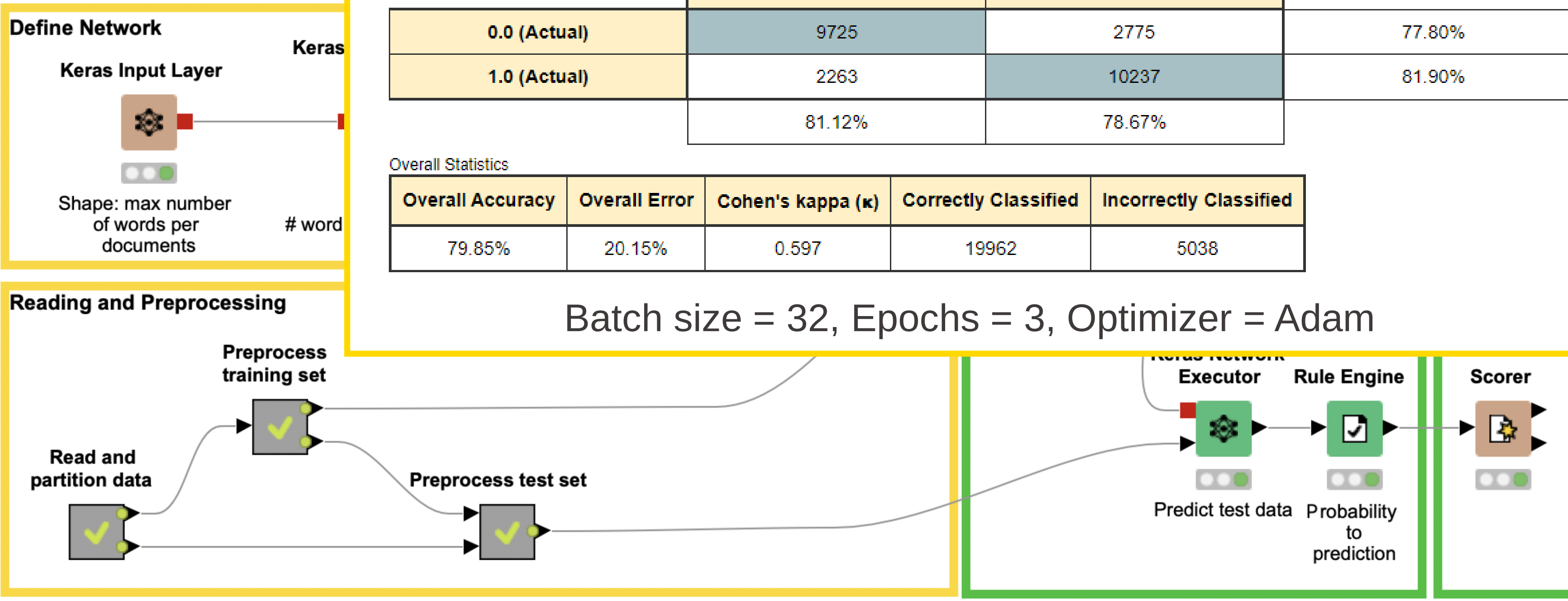
Network



Text Classification: Sentiment Analysis



Text Classification: Sentiment Analysis



Scorer View

Confusion Matrix

	0.0 (Predicted)	1.0 (Predicted)	
0.0 (Actual)	9725	2775	77.80%
1.0 (Actual)	2263	10237	81.90%
	81.12%	78.67%	

Overall Statistics

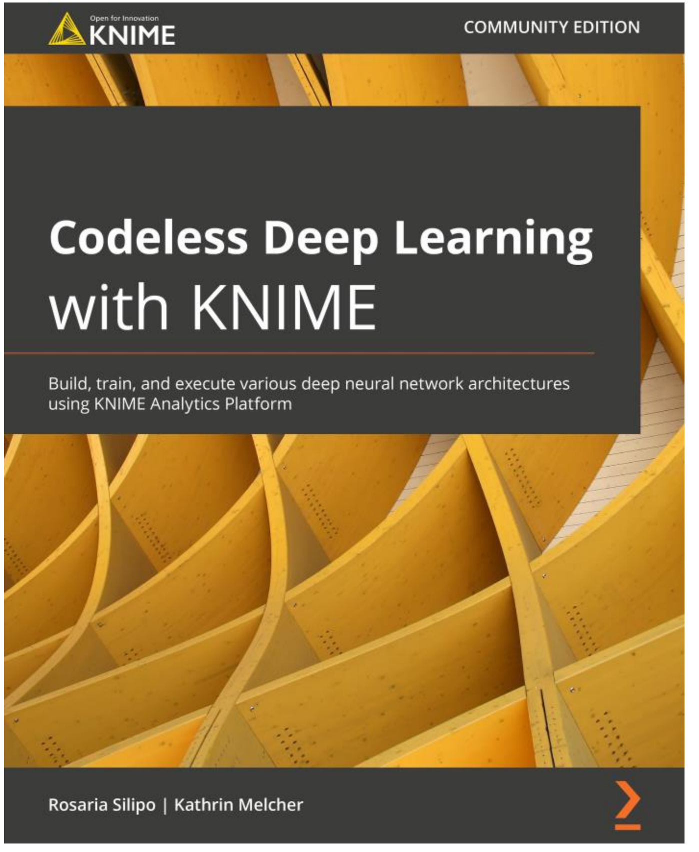
Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
79.85%	20.15%	0.597	19962	5038

Batch size = 32, Epochs = 3, Optimizer = Adam

Download the workflow from the [KNIME Hub]

To learn more...

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



Upcoming Online Courses

- Introduction to Text Processing
  - Nov 28 – Dec 2, 10-11:30am CST
  - https://www.knime.com/events/introduction-text-processing-2211

- Introduction to Time Series Analysis
  - Nov 29 – Dec 2, 10-11:30am CST
  - https://www.knime.com/events/introduction-time-series-analysis-2211

Use Code Joinus10 to get 10% discount on your registration

Thank you!

This slide deck is available at
https://kni.me/s/4-i5-EcLrZwf5cxN