Developed by Ioannis Konstantinidis

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

HPE DSI 312
Introduction to Deep Learning

Spring 2023

Instructor: Ioannis Konstantinidis

Overview

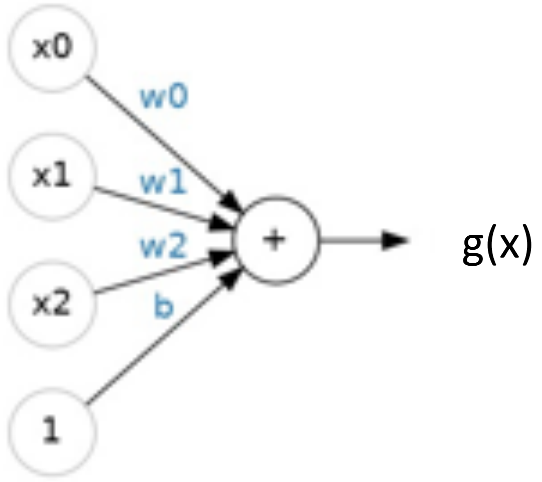
Neural networks:

- TensorFlow/Keras vs. Sklearn
- Transfer Learning
- Cross-entropy

Quick review

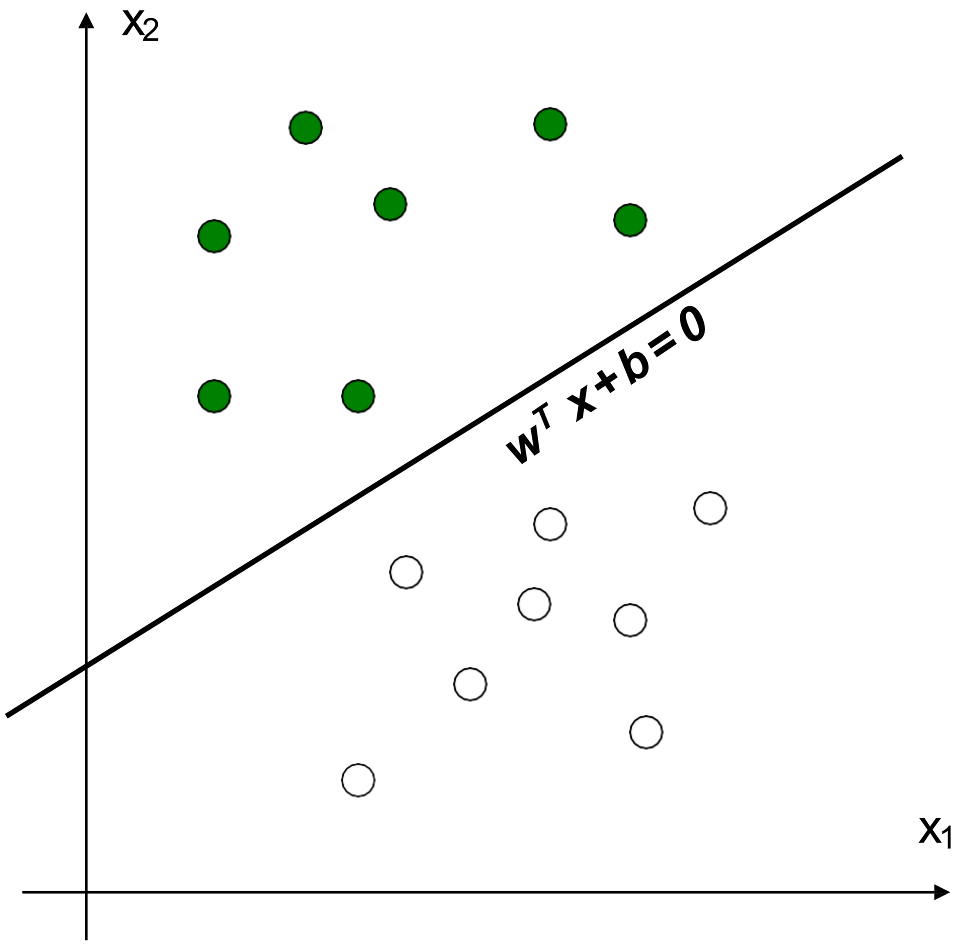
- Neural Networks
- Linear Units
- Perceptrons

Linear unit vs. linear regression: same form

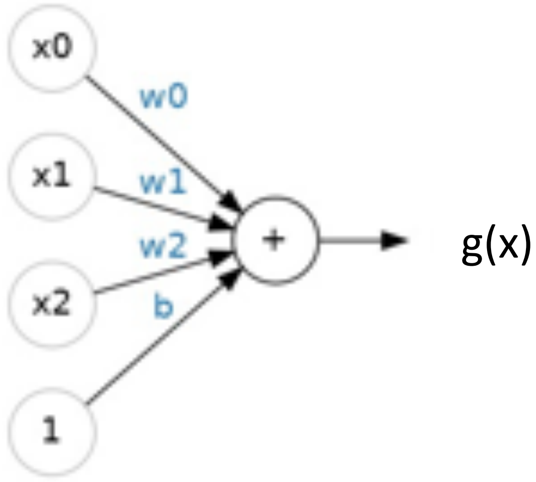


$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b =$$

$$= \sum_i w_i\, x_i + b$$

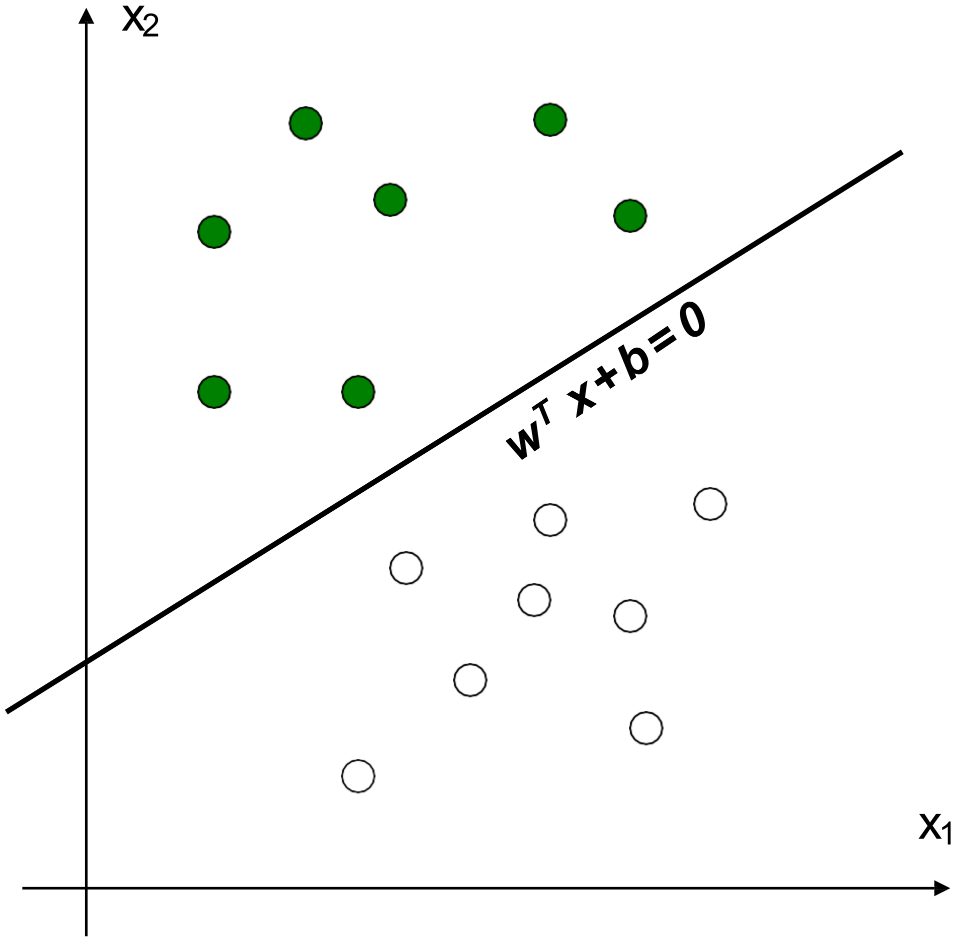


Linear unit vs. linear regression: different training



$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b =$$

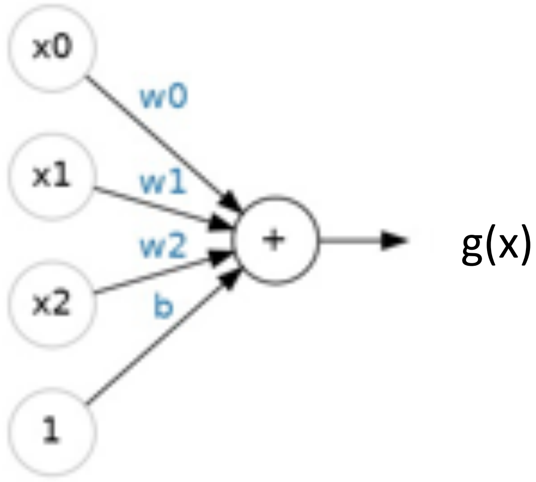
$$= \sum_i w_i \, x_i + b$$



Compute parameters (coefficients) via Ordinary Least Squares

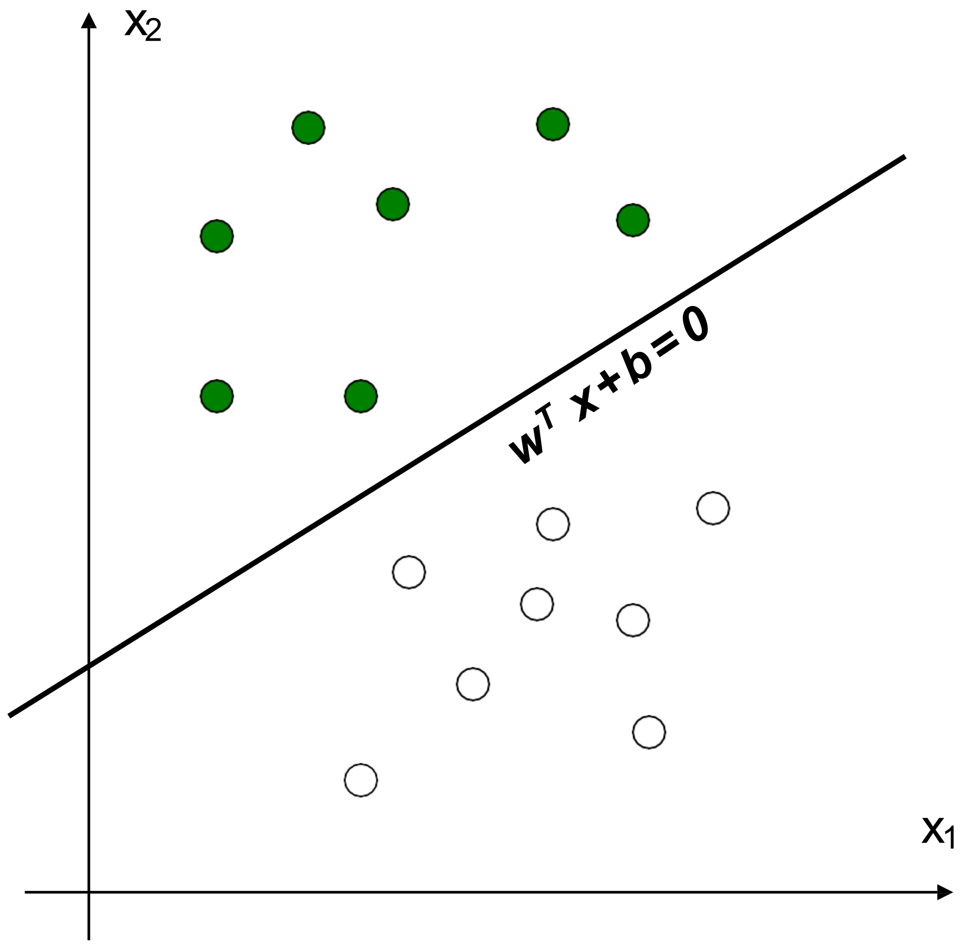
$$[\mathbf{W}, b] = (X^{\top} X)^{-1} X^{\top} Y$$

Linear unit vs. linear regression: different training



$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b =$$

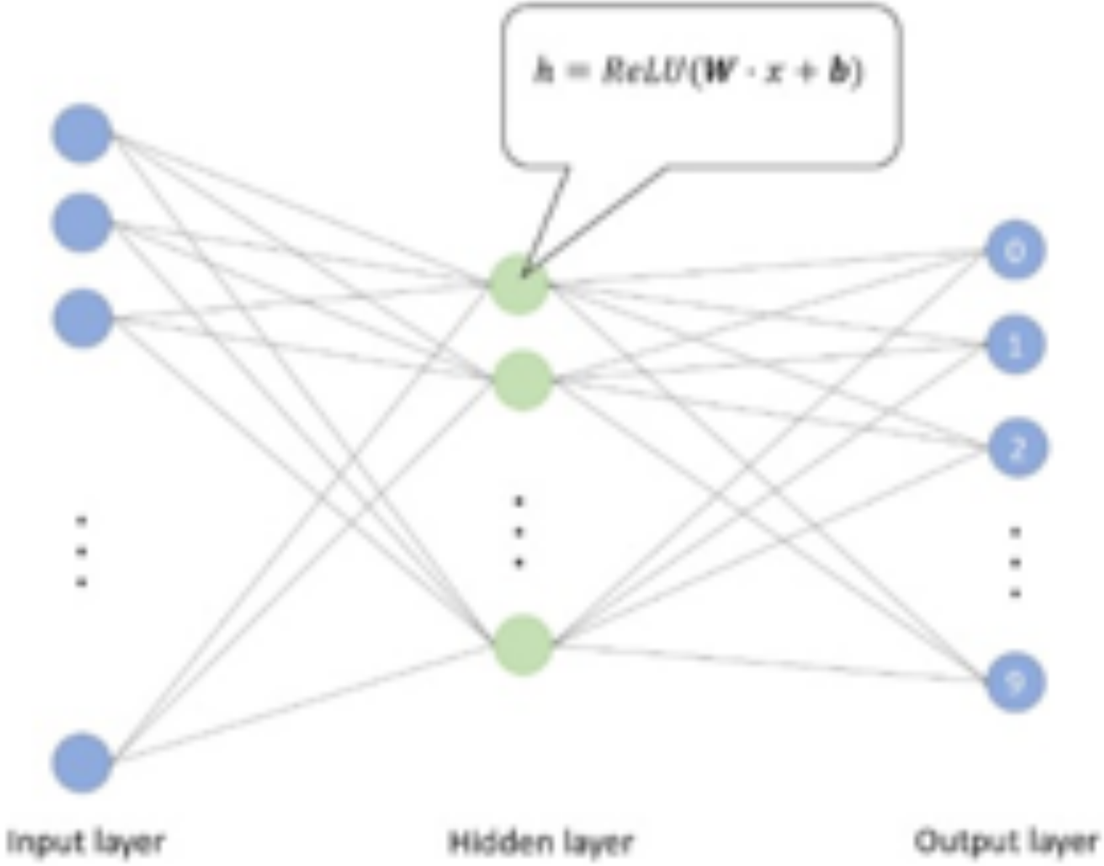
$$= \sum_i w_i \, x_i + b$$



Compute parameters (weights)
via backpropagation (SGD)

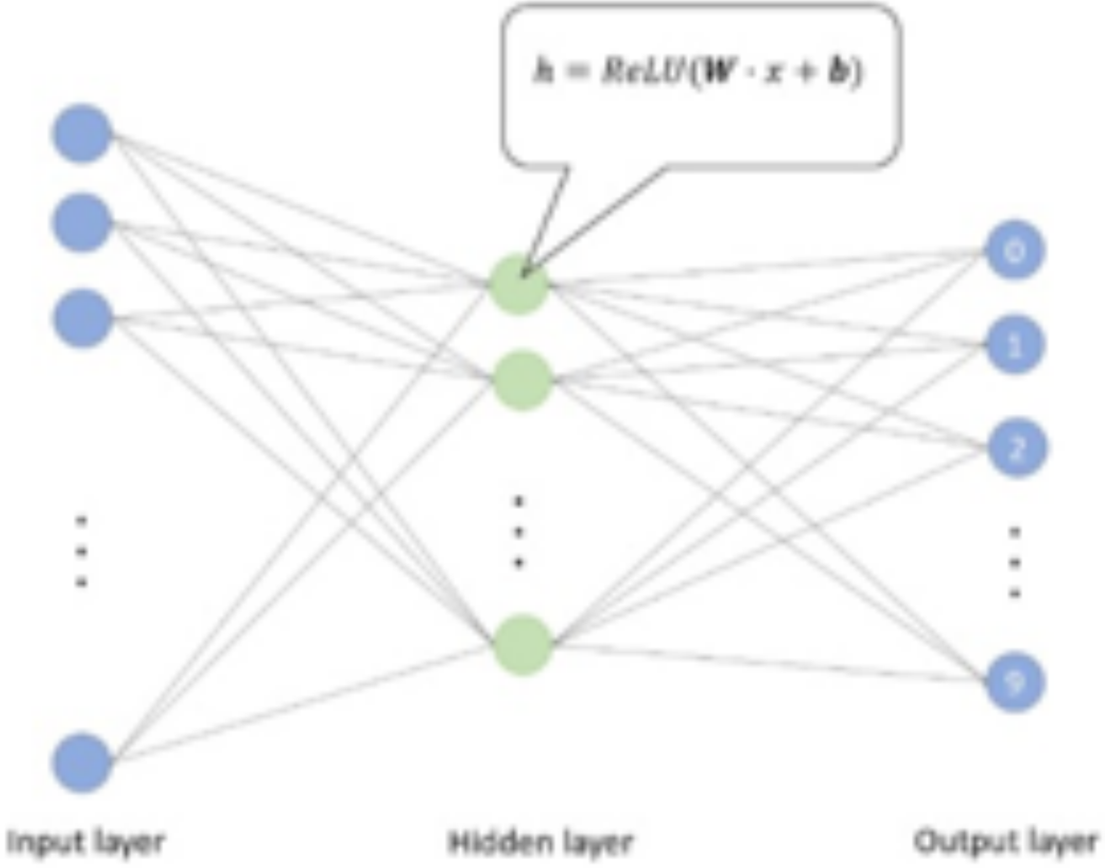
Network Architecture

Activation Functions: going beyond linear

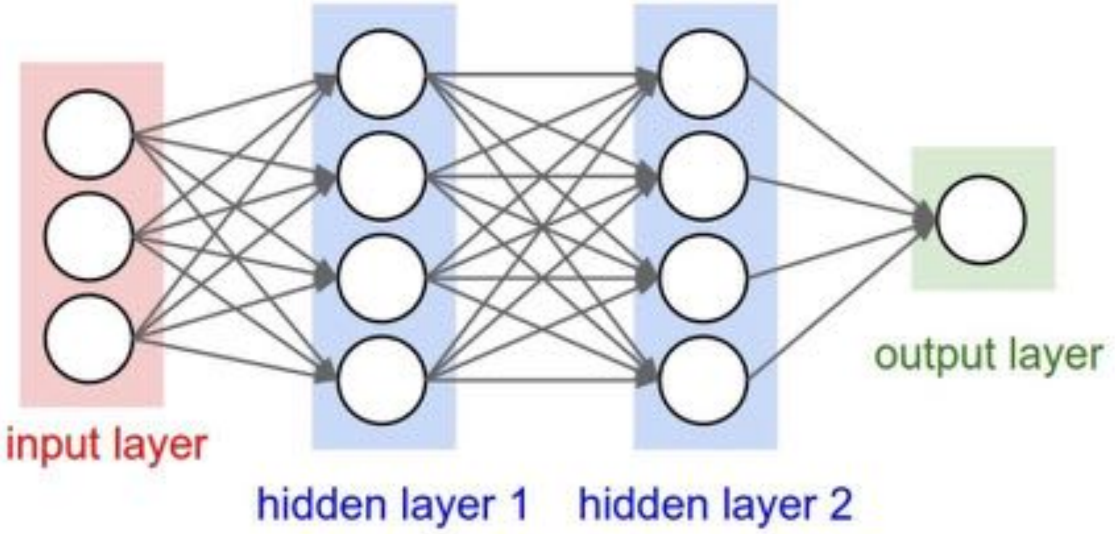


Network Architecture

Activation Functions: going beyond linear



Stacked layers: number and size

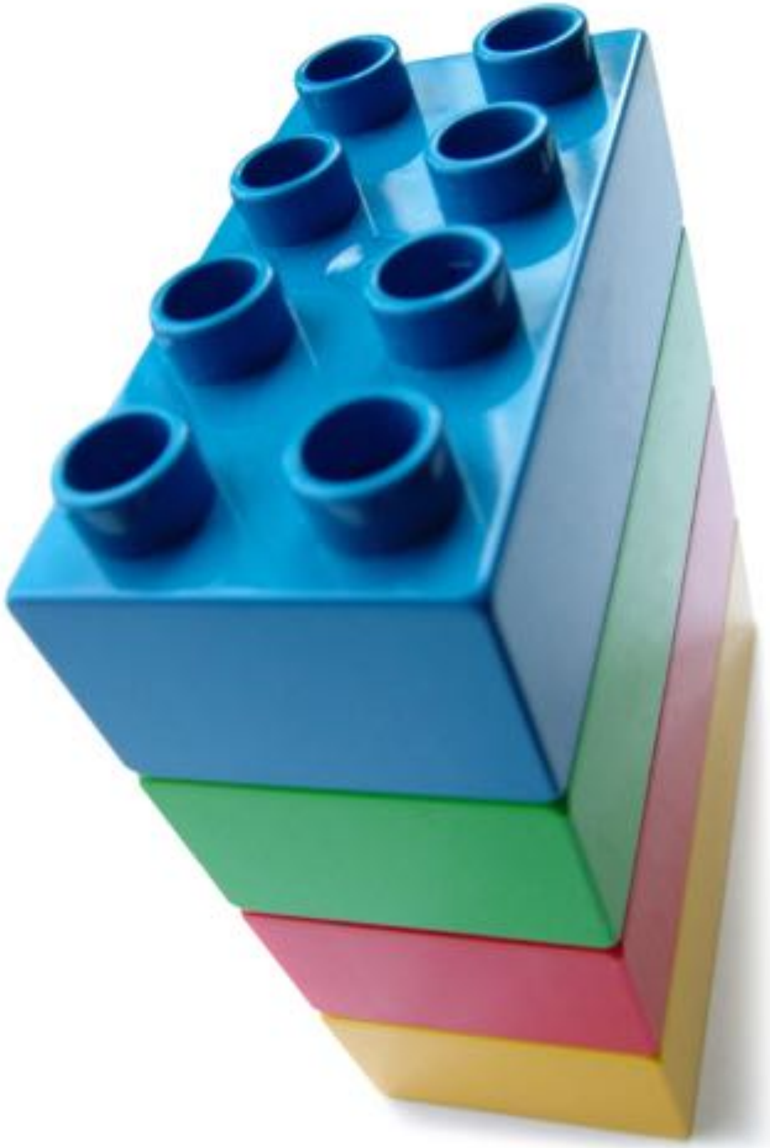


Perceptron example

- https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/anatomy
- https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises

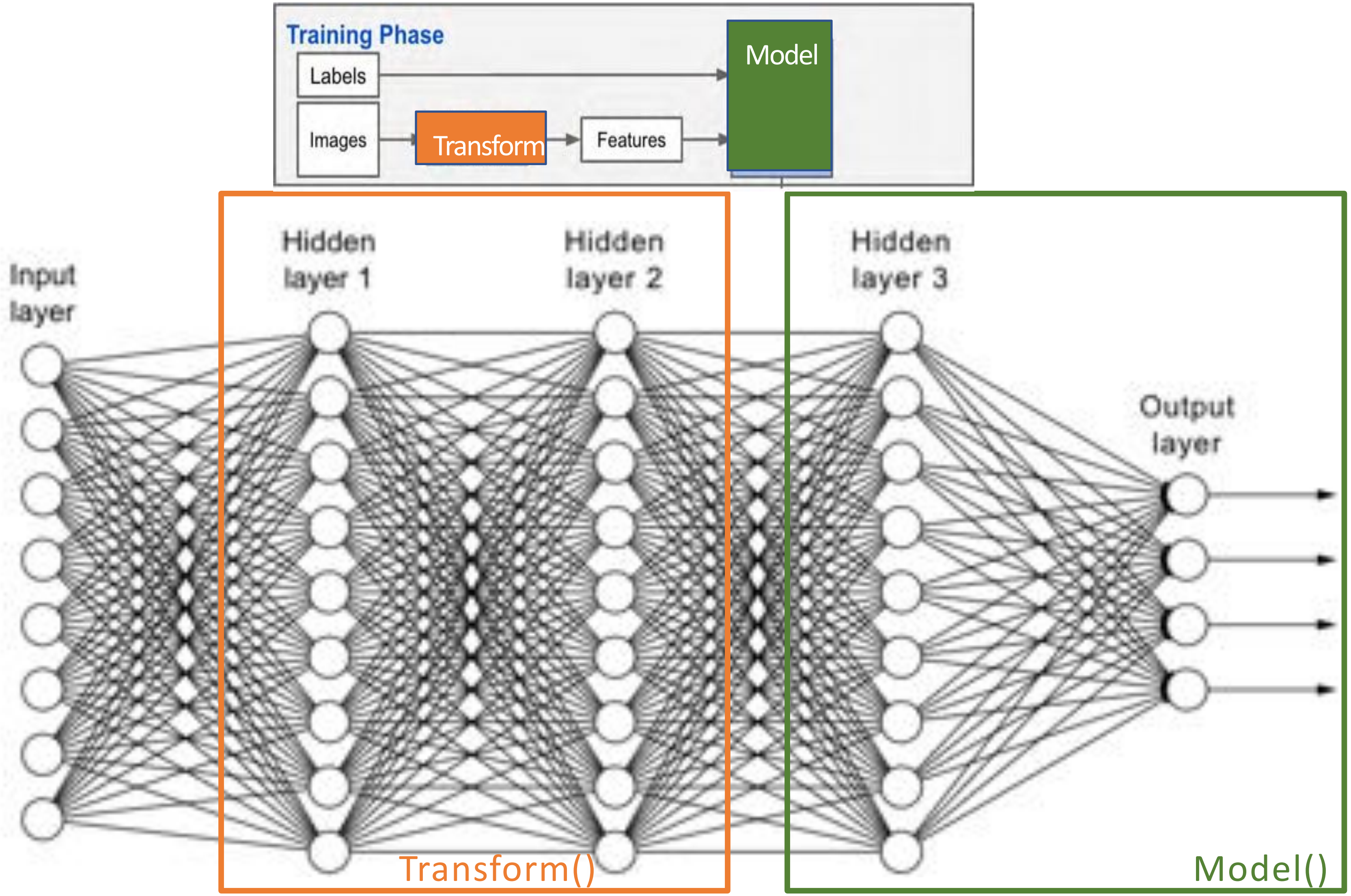
Transfer Learning

DL models are stacked into a complete pipeline



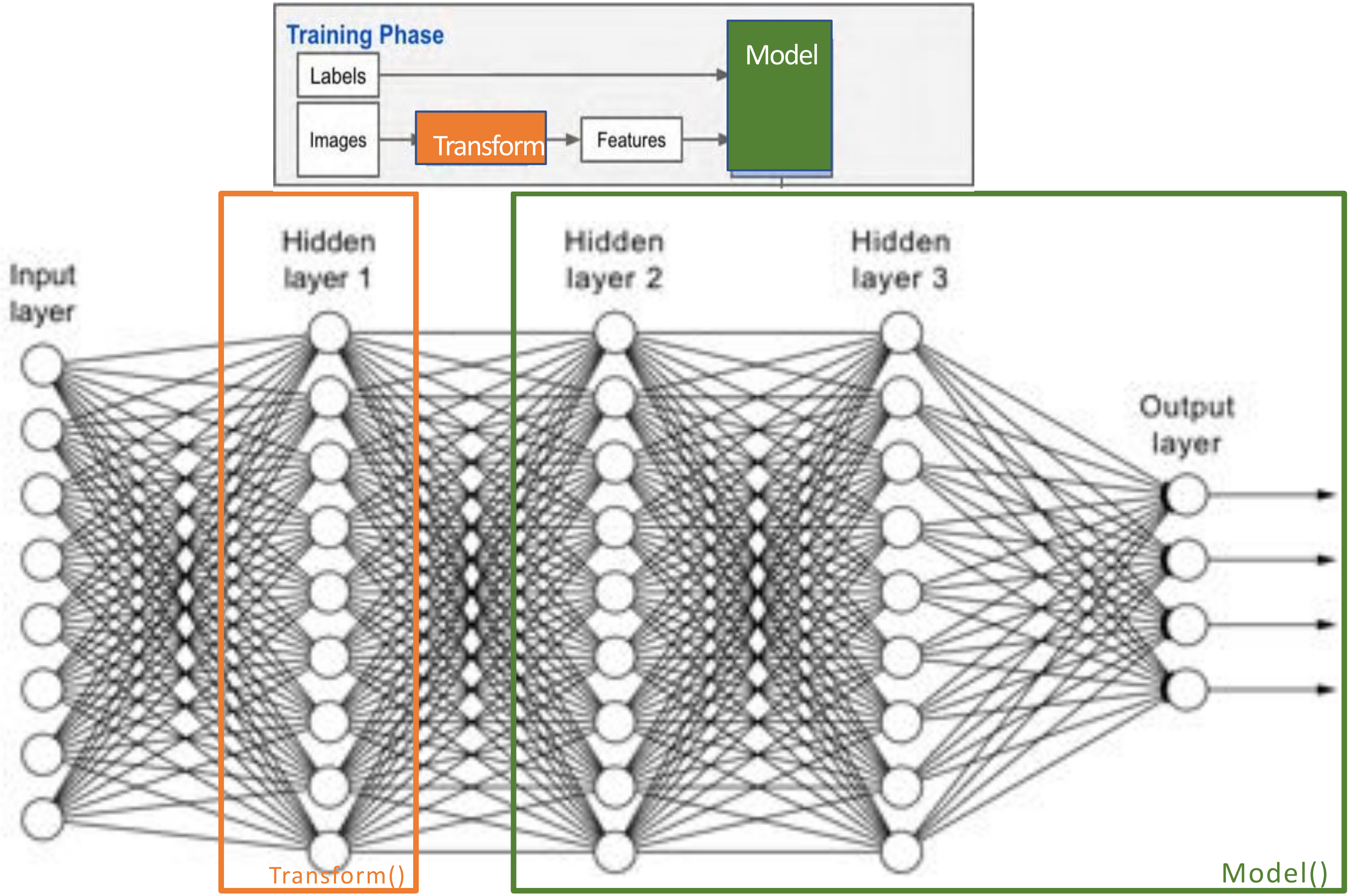
Make_pipeline(Transform,Model)

Recall:



Make_pipeline(Transform,Model)

Recall:



DL models are a complete pipeline

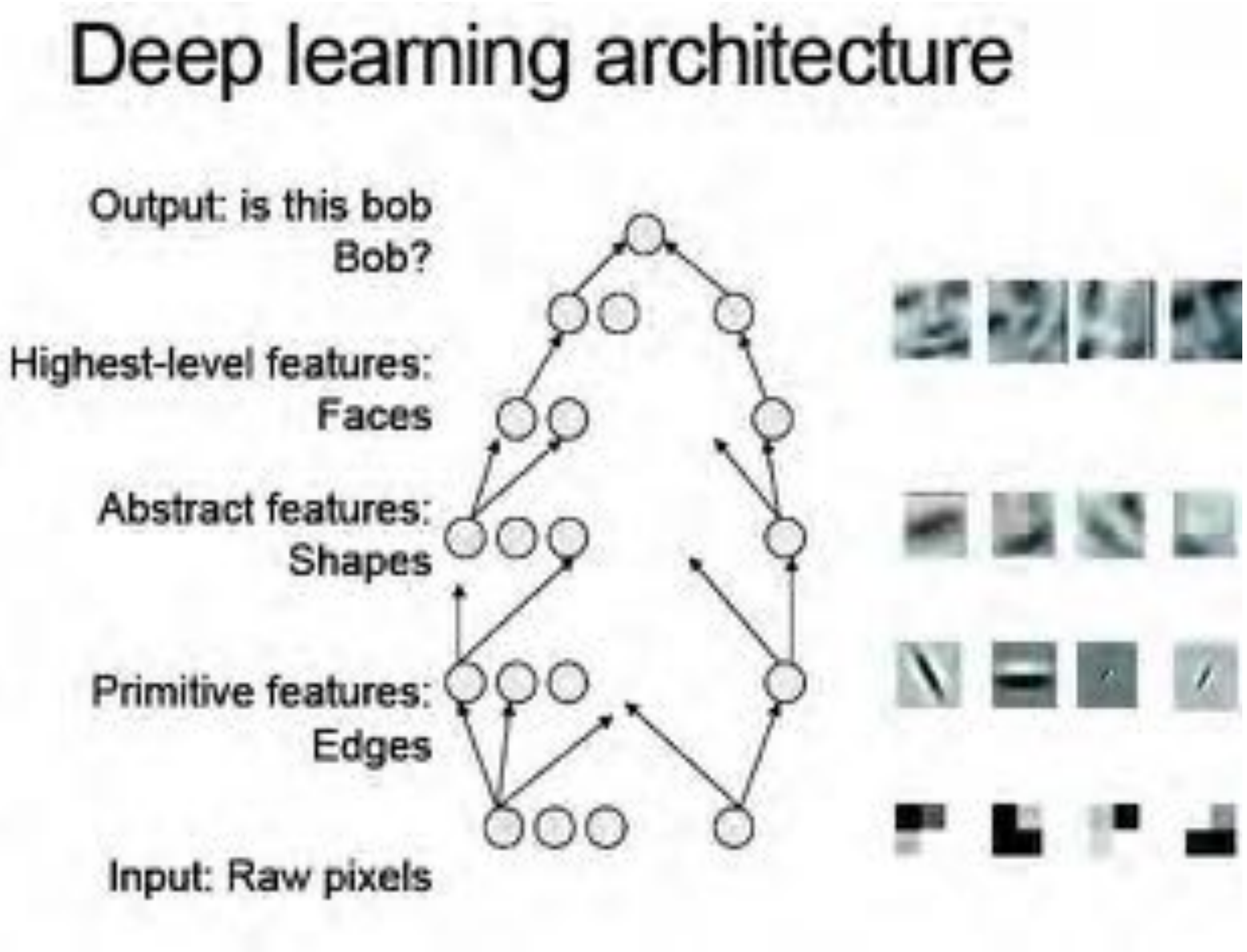
- Feature engineering is automatically “baked into” the process
- Initial layers pick out “low level” features
- Later layers process these transformed data to compute ”higher level” features
- “Top” layers perform classification tasks based on these custom-designed features

Example from Computer Vision

“Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features.

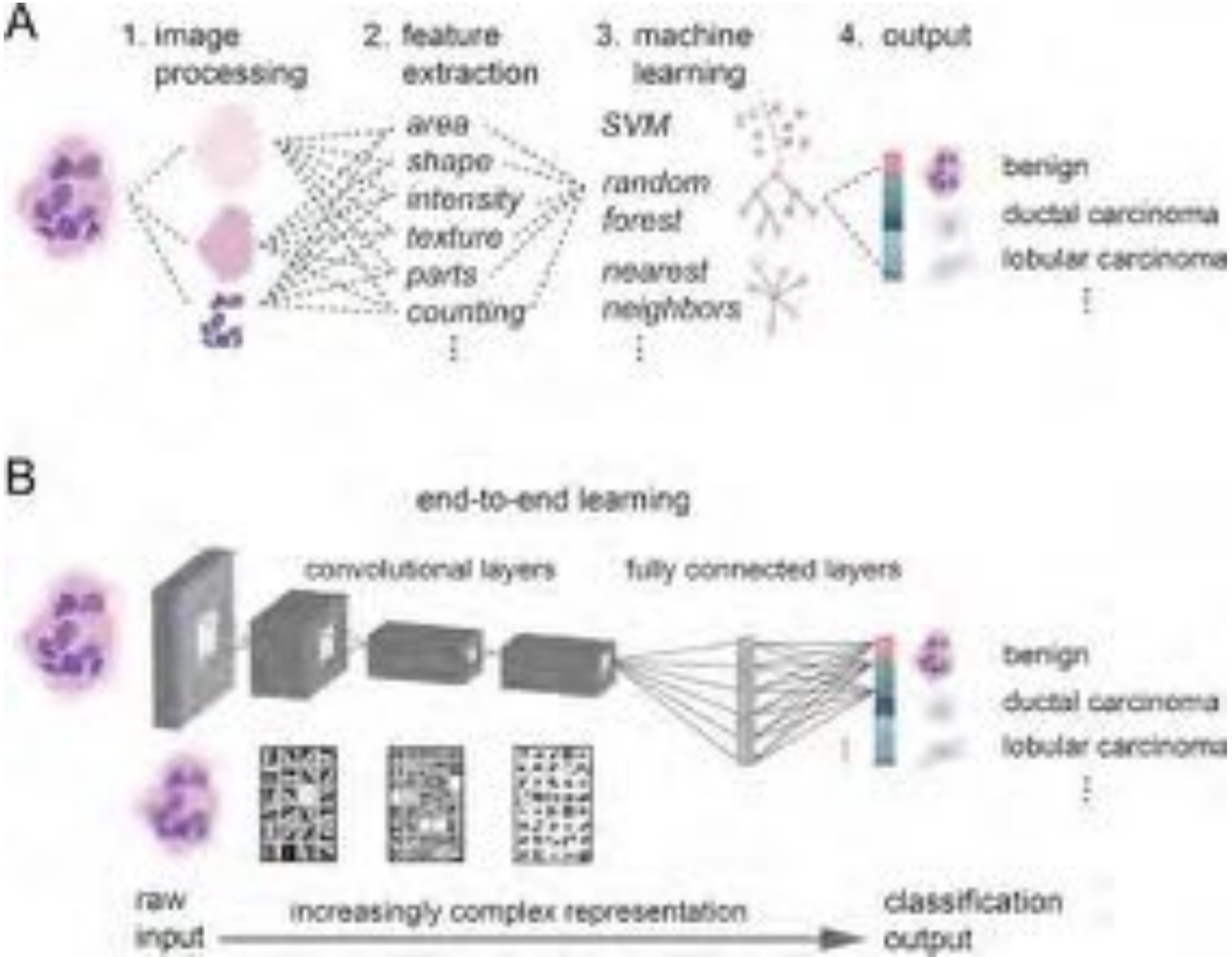
Automatically learning features at multiple levels of abstraction allows a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.”

[Bengio, “On the expressive power of deep architectures”, Talkat ALT, 2011]
[Bengio, Learning DeepArchitectures for AI, 2009]



Earlier ML

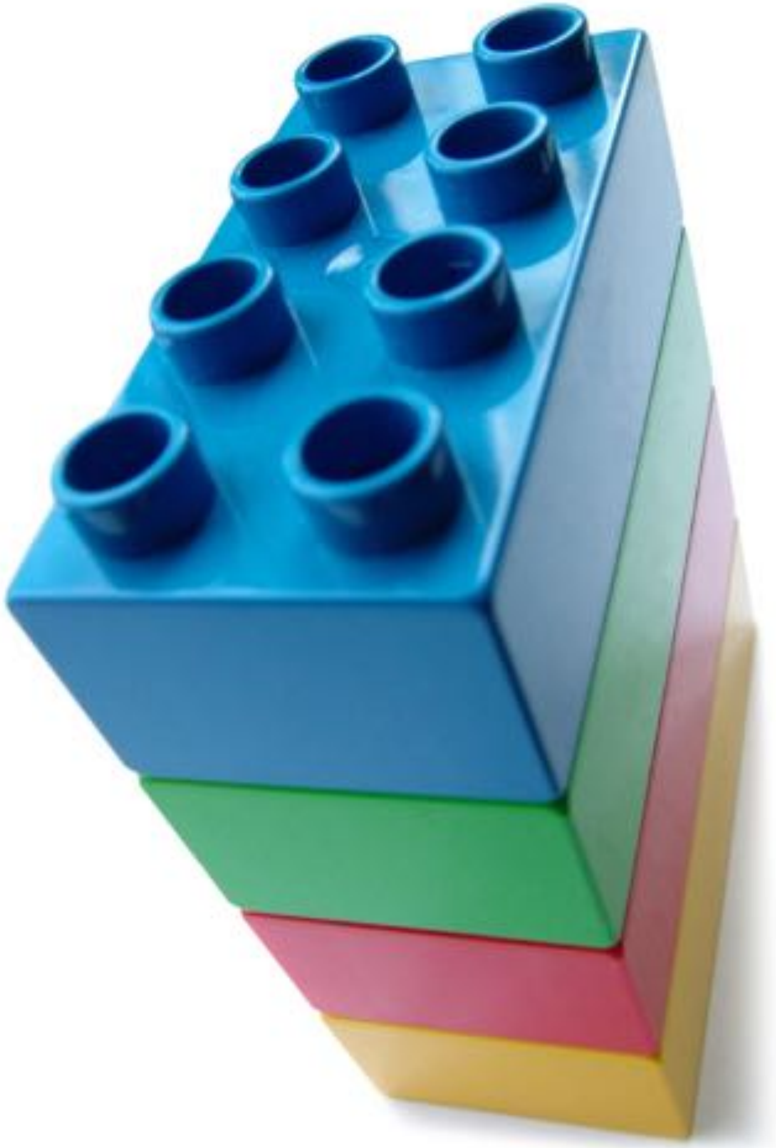
Deep Learning



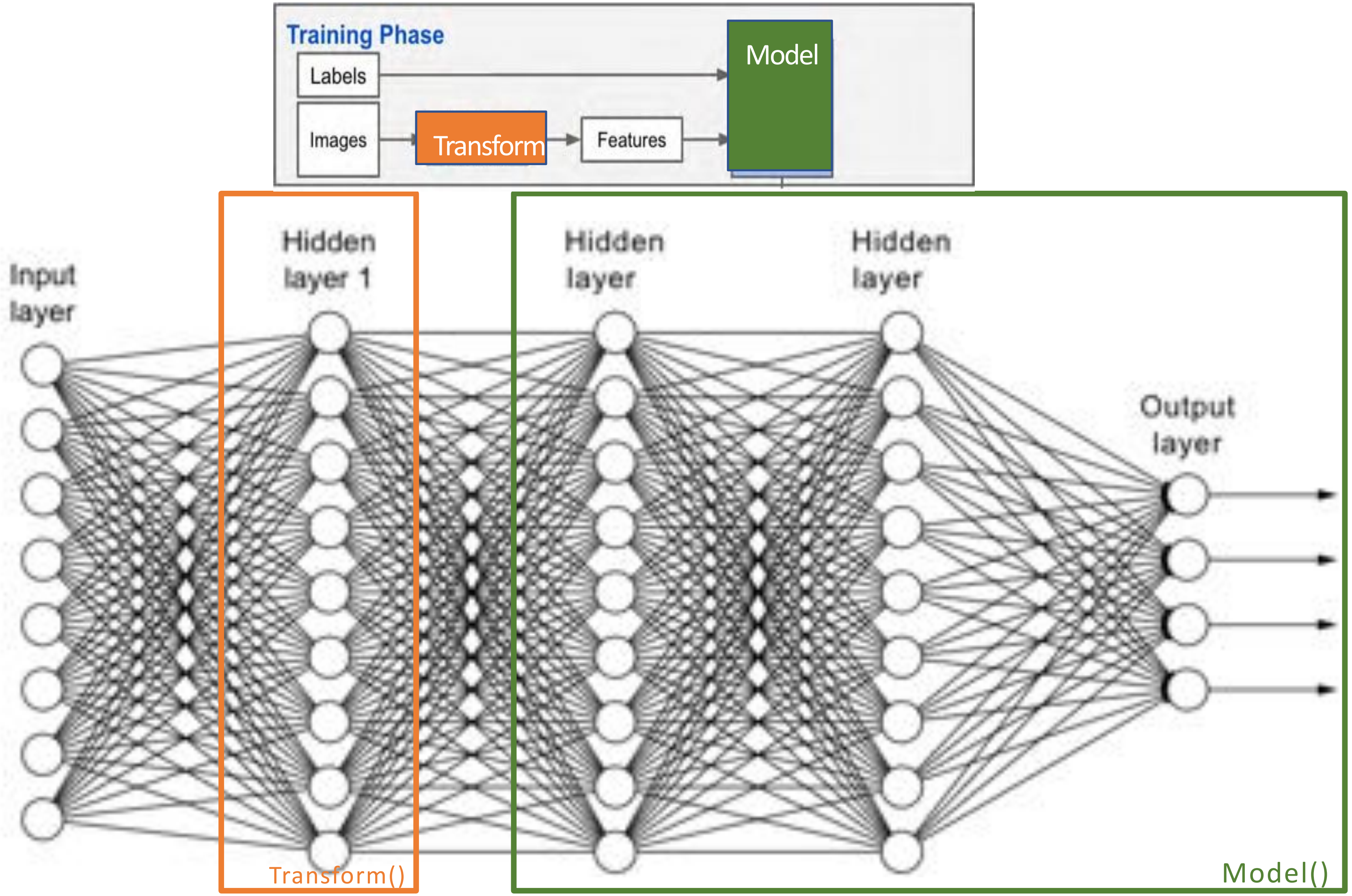
Transfer Learning

DL models are stacked into a complete pipeline

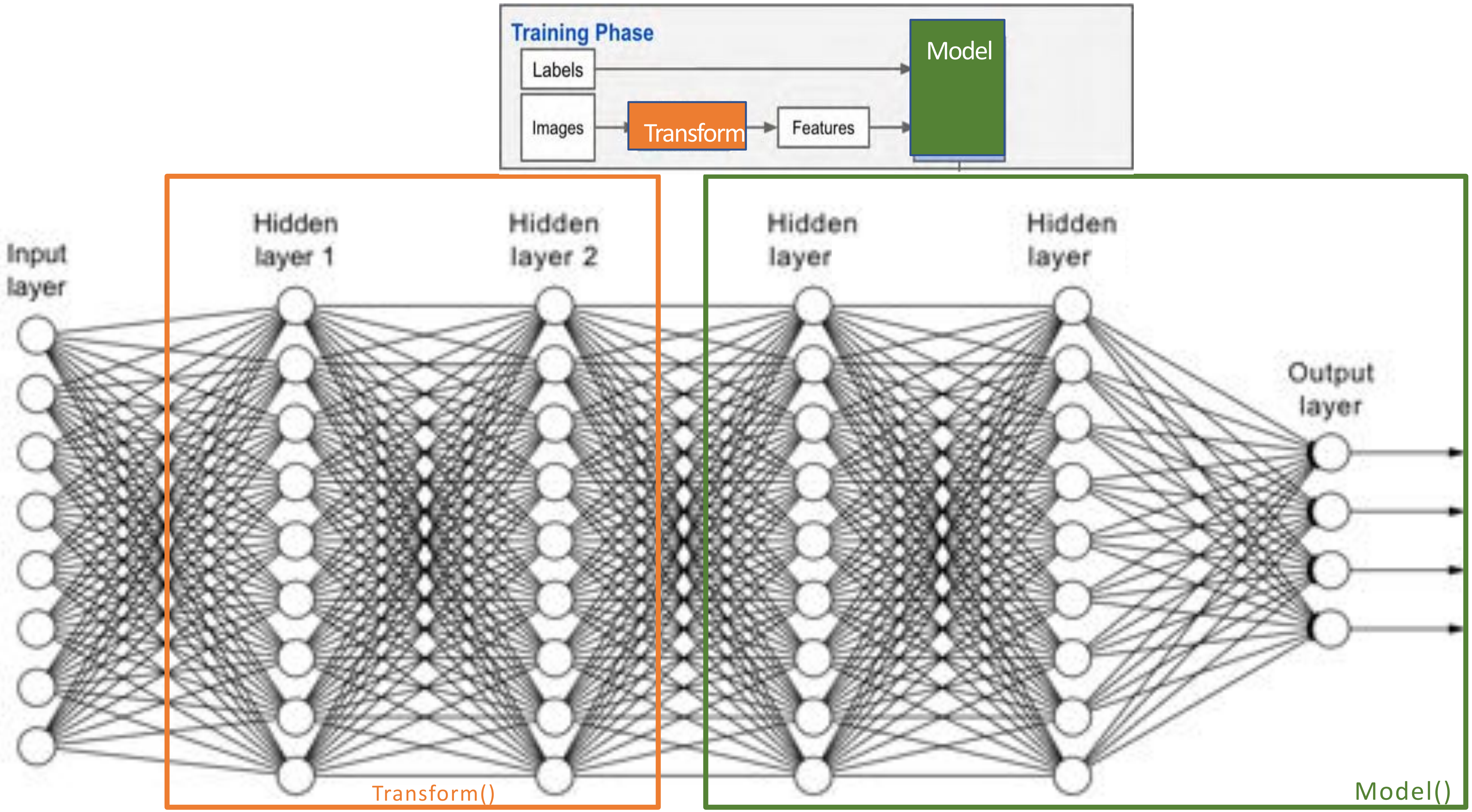
DL models are modular stacks



Make_pipeline(Transform,Model)



Make_pipeline(Transform,Model)



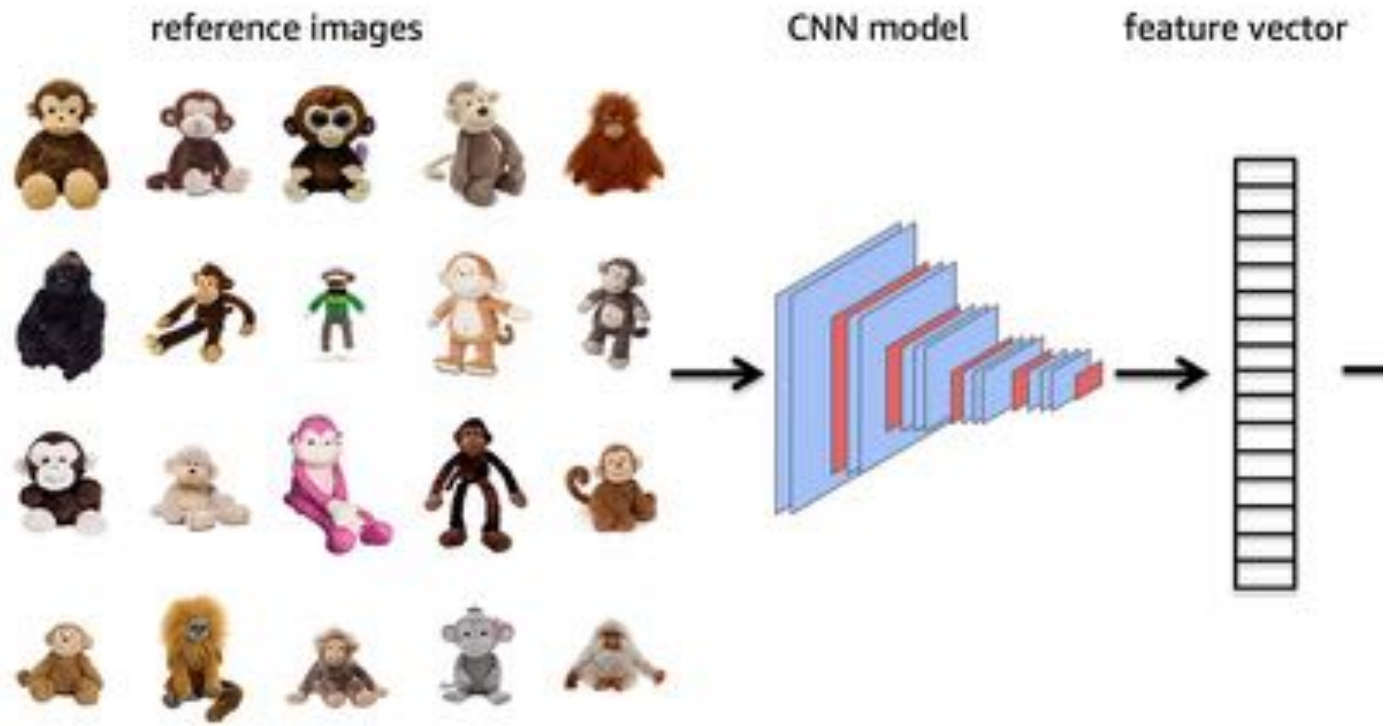
Transfer Learning

Training DL models from scratch is expensive

Transfer learning takes a piece of a model that has already been trained on a related task and reuses it in a new model

Data to features module

https://www.tensorflow.org/hub/common_saved_model_apis/images

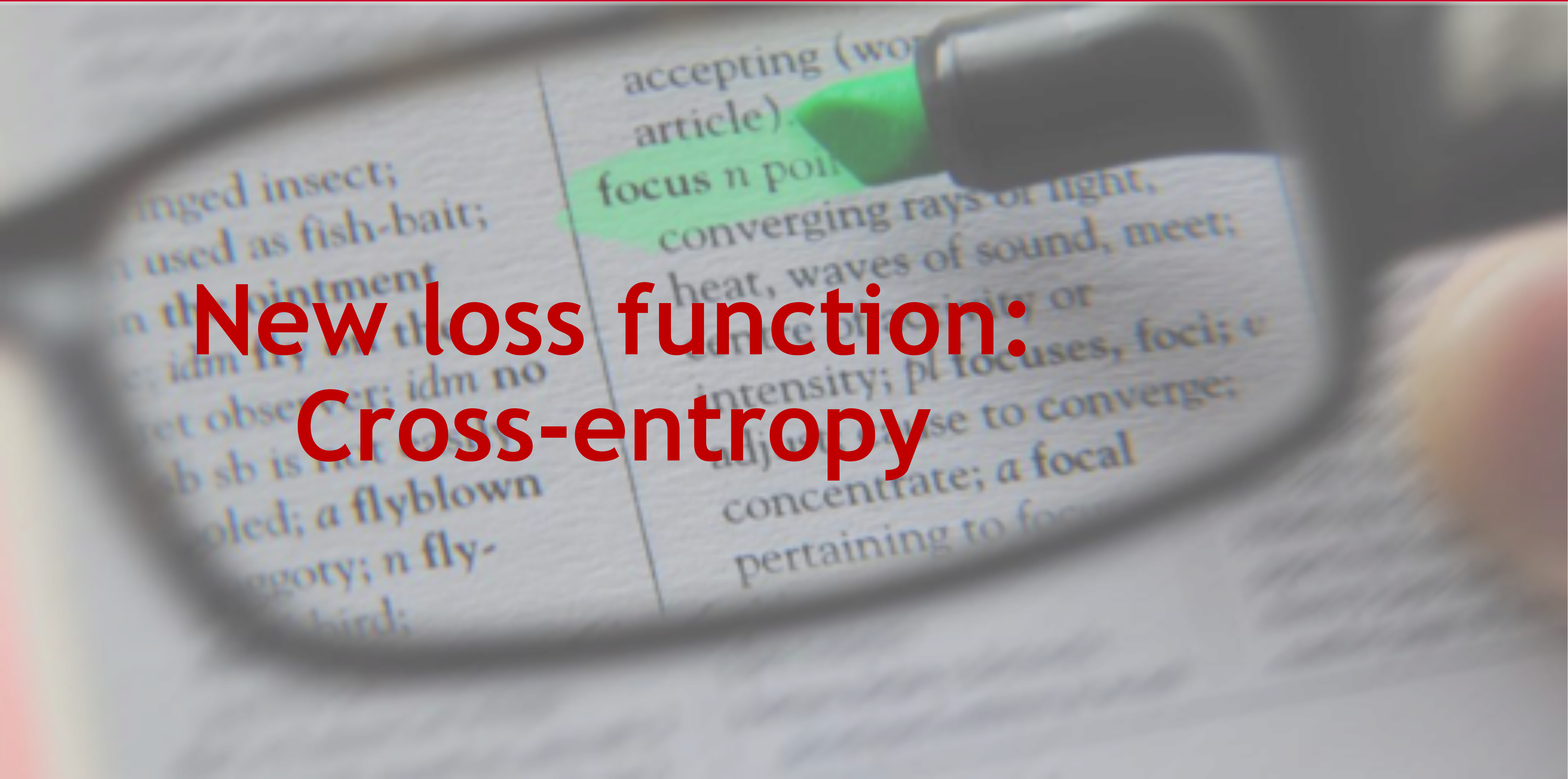


Actual demo (part I)

- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.

  https://tensorflow.org/hub/tutorials/tf2_image_retraining

- We will revisit this in more detail next time



New loss function: Cross-entropy

loss=tf.keras.losses.

- Actual class value for label y

binary choice: 0 or 1

- Prediction for label y

probability: p

loss=tf.keras.losses.

- Actual class value for label y

binary choice: 0 or 1

- Prediction for label y

probability: p

- L^2 loss

$$(y - p)^2$$

or

$$(y - \mathrm{round}(p))^2$$

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

L^2 loss / squared error

$$(y - p)^2$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	$(0-0.01)^2 = 0.0001$	$(0-0.1)^2 = 0.01$	$(0-1)^2 = 1$
Y=1	$(1-0.01)^2 = 0.99^2=0.98$	$(1-0.1)^2 = 0.9^2=0.81$	$(1-1)^2 = 0^2$

$$(y - \text{round}(p))^2$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	$(0-0)^2 = 0$	$(0-0)^2 = 0$	$(0-1)^2 = 1$
Y=1	$(1-0)^2 = 1$	$(1-0)^2 = 1$	$(1-1)^2 = 0$

L^2 loss / squared error

$$(y - p)^2$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	= ~0	= ~0	= 1
Y=1	= ~1	= ~1	= 0

$$(y - \mathrm{round}(p))^2$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	= 0	= 0	= 1
Y=1	= 1	= 1	= 0

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

loss=tf.keras.losses.

- Actual class value for label y

binary choice: 0 or 1

- Prediction for label y

probability: p

- L^2 loss

$$(y - p)^2$$

or

$$(y - \mathrm{round}(p))^2$$

- Surprise factor

$$y\log(p) + (1 - y)\log(1 - p)$$

Surprise factor: $y\log(p) + (1 - y)\log(1 - p)$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	-(1-0)*log(1-0.01) = -log(0.99) = ~0.01	-(1-0)*log(1-0.1) = -log(0.9) = ~0.1	-(1-0) * log(1-1) = -log(0) = ~infinity
Y=1	-1 * log(0.01) = ~4.6	-1 * log(0.1) = ~2.3	-1 * log(1) = 0

Surprise factor: $y\log(p) + (1 - y)\log(1 - p)$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	= ~0.01 No surprise	= ~0.1 Almost no surprise	= ~infinity SURPRISE !!!!
Y=1	= ~4.6 SURPRISE	= ~2.3 Surprise	= 0 Absolutely no surprise

Surprise factor vs. squared error

$$y\log(p) + (1-y)\log(1-p)$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	= ~0.01 No surprise	= ~0.1 Almost no surprise	= ~infinity SURPRISE !!!!
Y=1	= ~4.6 SURPRISE	= ~2.3 Surprise	= 0 Absolutely no surprise

$$(y - \mathrm{round}(p))^2$$

Actual class label	Prob(Y=1) = 0.01	Prob(Y=1) = 0.1	Prob(Y=1) = 1
Y=0	= 0	= 0	= 1
Y=1	= 1	= 1	= 0

Surprise factor, AKA

- Log loss
- Likelihood Function
- Binary/Categorical (relative) Cross-Entropy
- Kullback–Leibler divergence