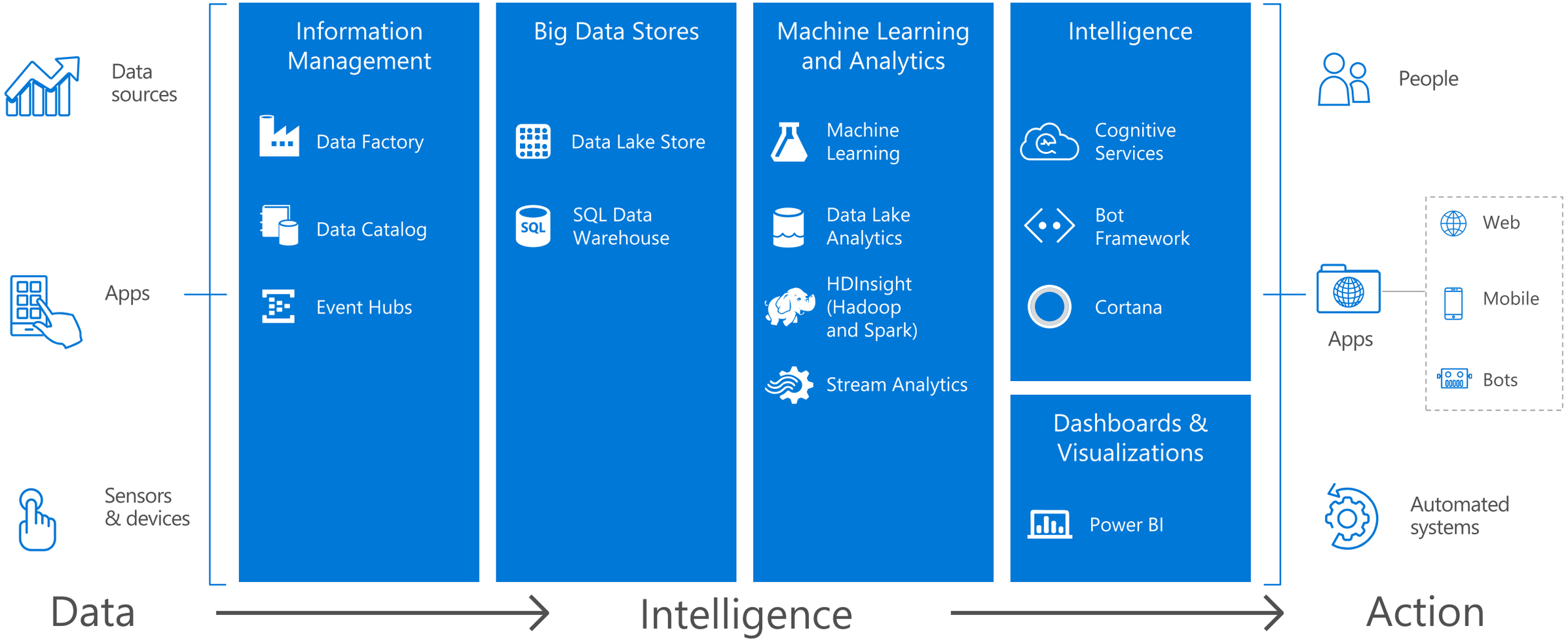
Azure Machine Learning

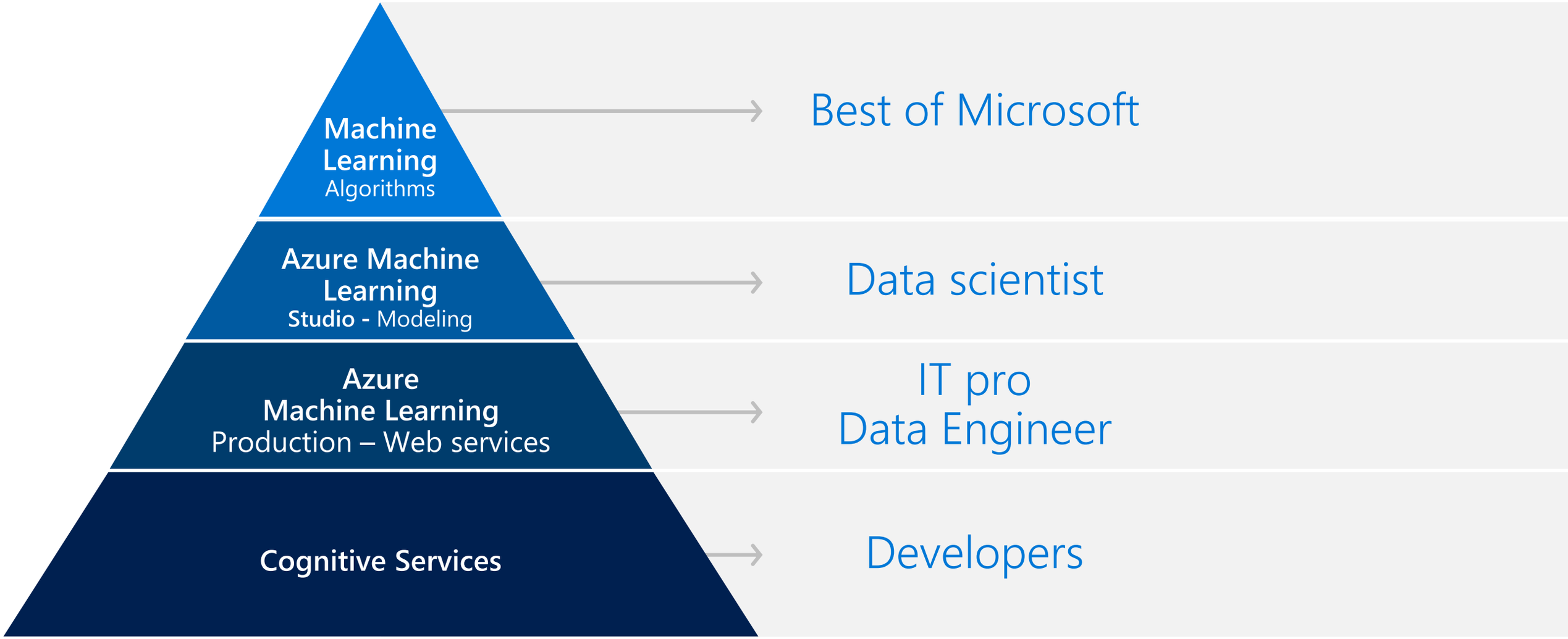
Manjunath S

Microsoft

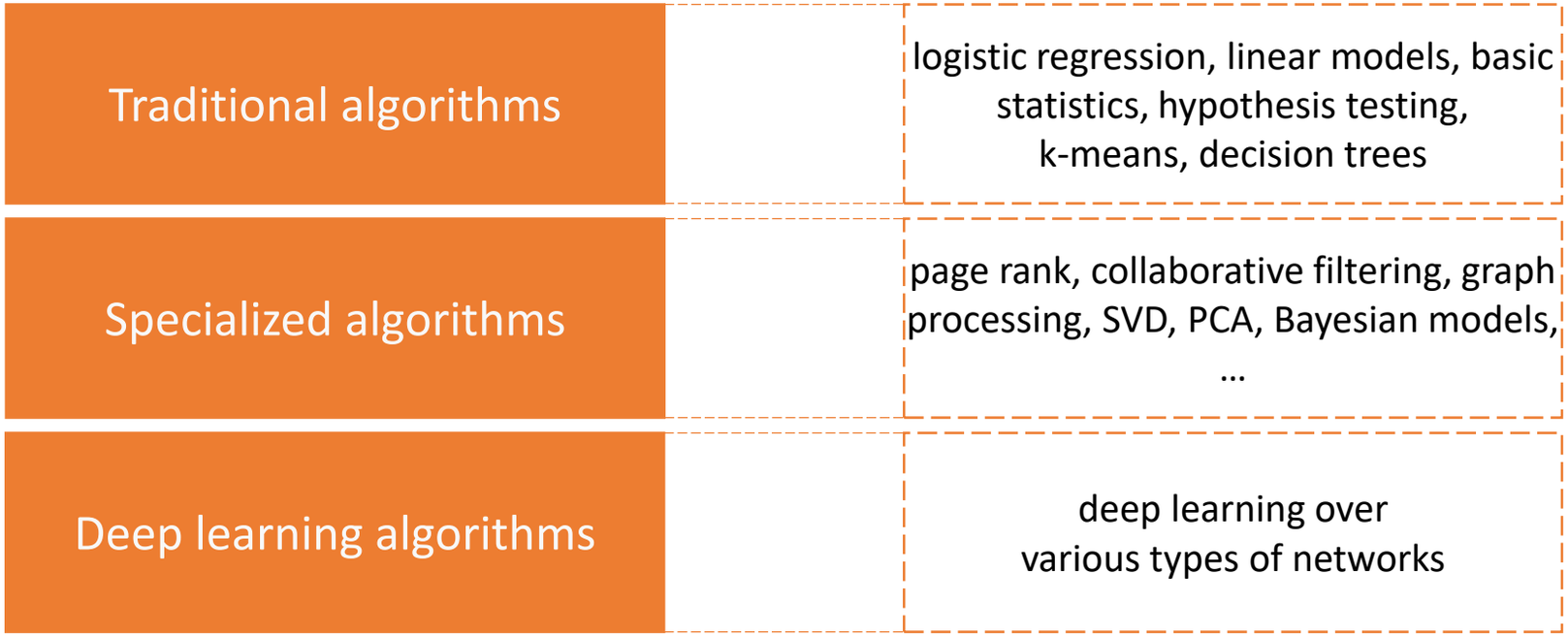
Cortana Intelligence Suite Services



Machine Learning services in the cloud

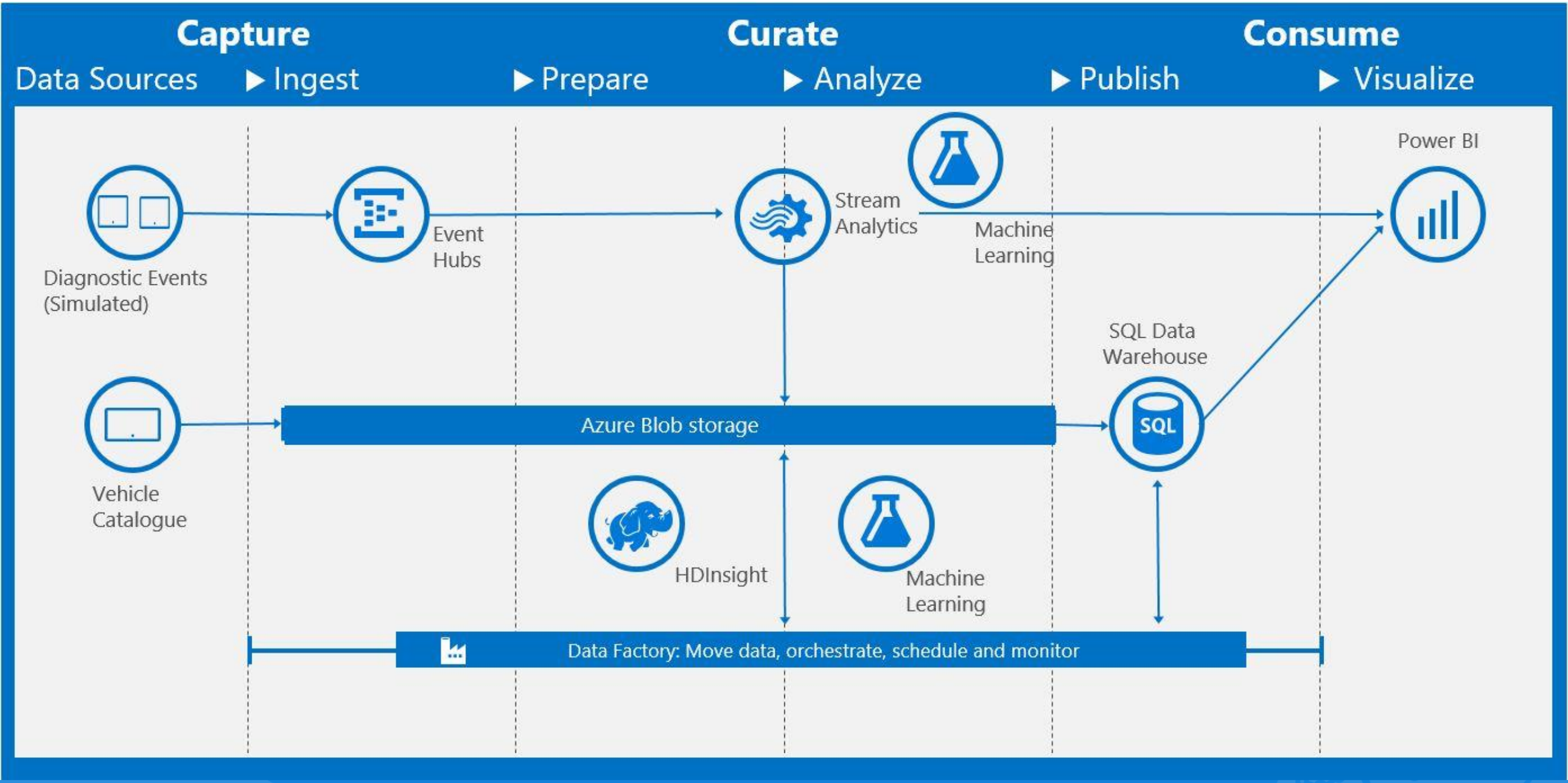


Infinite world of scalable machine learning

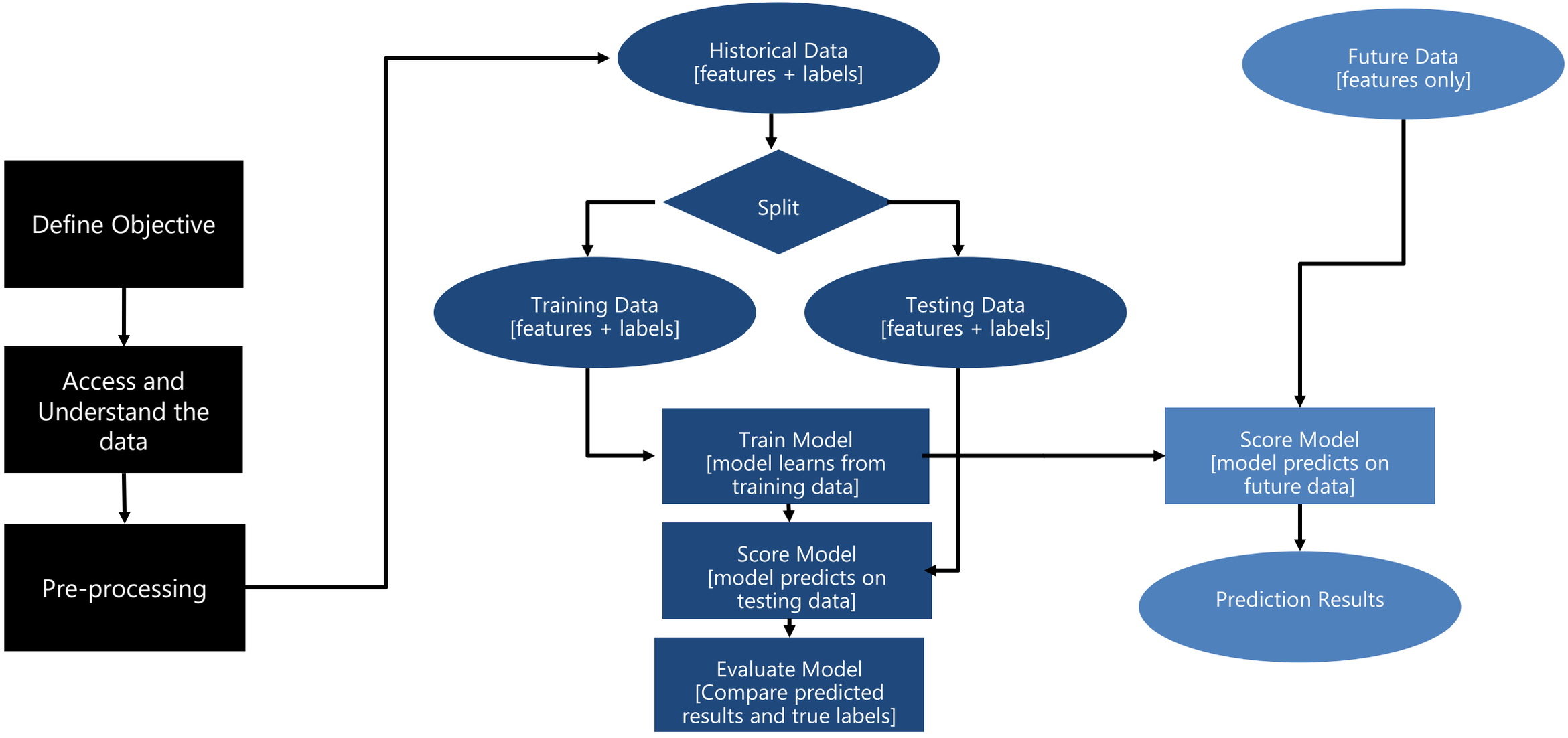


Use cases of scalable machine learning

	Retail	Financial services	Healthcare	Manufacturing
Traditional algorithms	loyalty programs customer acquisition pricing strategy supply chain mgnt	customer churn fraud detection risk & compliance cross-sell & upsell personalization	bill collection operational efficiency patient demographics pay for performance	demand forecasting pricing strategy supply chain optimization predictive maintenance remote monitoring
Specialized algorithms	product recommendations intelligent search routing robotics	ad placement predictive maintenance		
Deep learning algorithms	image, video recognition sentiment analysis text comprehension natural language processing	robotics bots augmented reality predictive maintenance		


Capture
Curate
Consume
Data Sources
Ingest
Prepare
Analyze
Publish
Visualize
Diagnostic Events (Simulated)
Event Hubs
Stream Analytics
Machine Learning
Power BI
SQL Data Warehouse
Vehicle Catalogue
Azure Blob storage
SQL
HDInsight
Machine Learning
Data Factory: Move data, orchestrate, schedule and monitor

Data Science Process



Deep learning at Microsoft

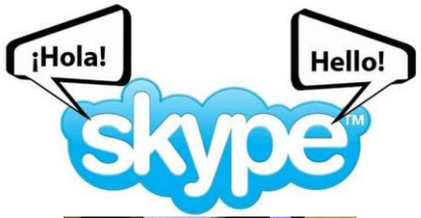
- Microsoft Cognitive Services
- Skype Translator
- Cortana
- Bing
- Bing Ads
- Augmented Reality
- Microsoft Research

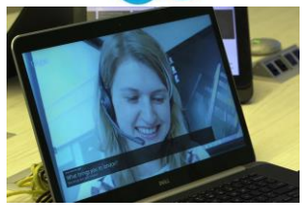
Cortana

Bing ads

Bing

XBOX





Microsoft Research

The Microsoft Cognitive Toolkit (CNTK)

- CNTK is Microsoft's **open-source**, **cross-platform** toolkit for learning and evaluating **deep neural networks**

- CNTK expresses (nearly) **arbitrary neural networks** by composing simple building blocks into complex **computational networks**, supporting relevant network types and applications.

- CNTK is **production-ready**: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.

“CNTK is production-ready: State-of-the-art accuracy, efficient, and scales to multi-GPU/multi-server.”

Benchmarking on a single server by HKBU

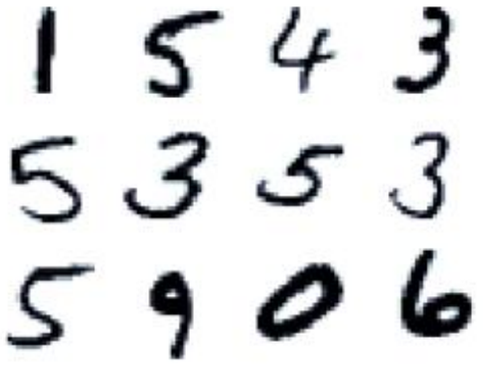
seconds per minibatch on G1080 (G980) GPU; lower=better

	FCN-8	AlexNet		ResNet-50		LSTM-64
CNTK	**0.037**	0.040	(0.054)	0.207	(0.245)	**0.122**
Caffe	0.038	**0.026**	**(0.033)**	0.307	(-)	-
TensorFlow	0.063	-	(0.058)	-	(0.346)	0.144
Torch	0.048	0.033	(0.038)	**0.188**	**(0.215)**	0.194

[“Benchmarking State-of-the-Art Deep Learning Software Tools,” http://arxiv.org/pdf/1608.07249v5.pdf]

MNIST Handwritten Digits (OCR)

Handwritten Digits

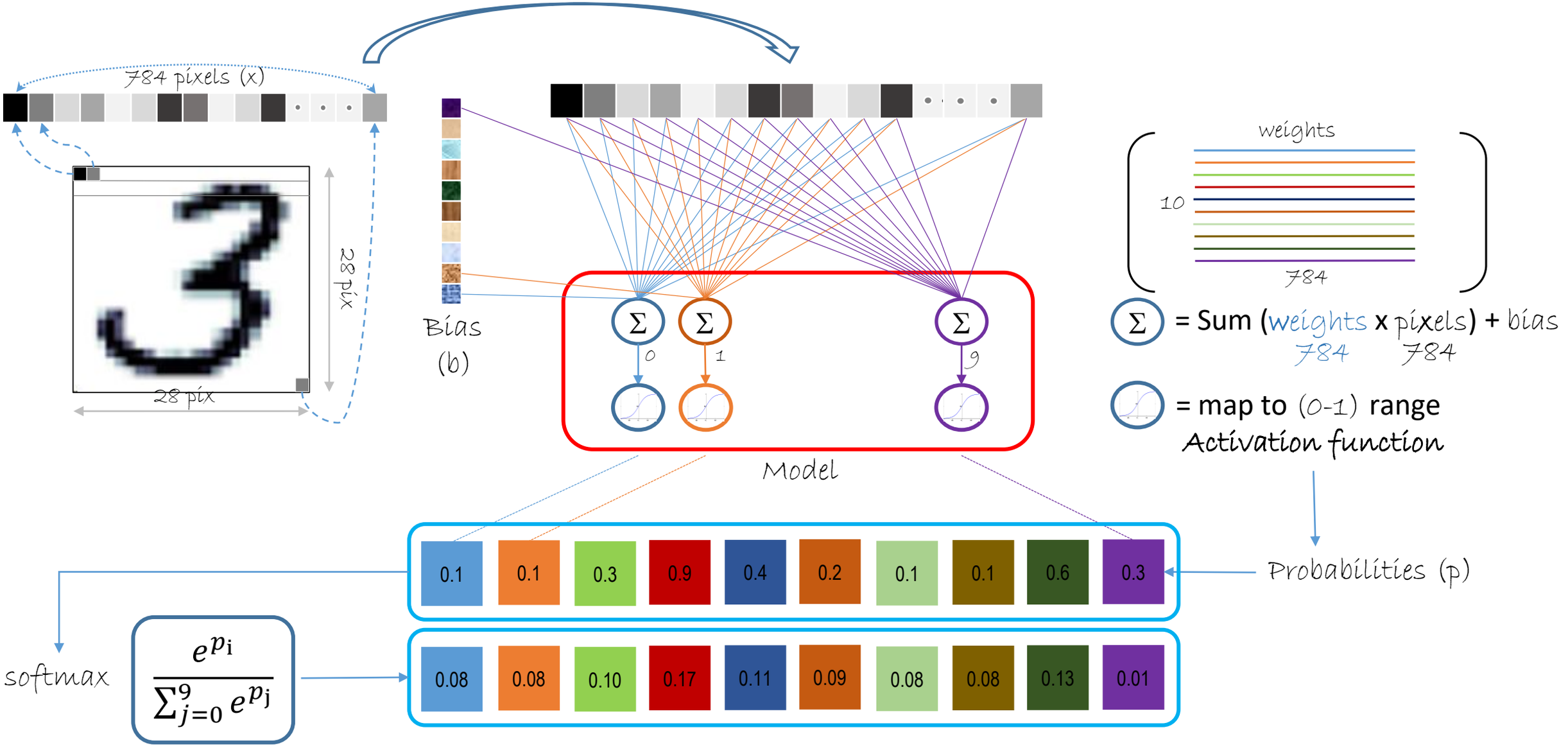


$$\left\{ \begin{array}{cccc} 1 & 5 & 4 & 3 \\ 5 & 3 & 5 & 3 \\ 5 & 9 & 0 & 6 \end{array} \right\}$$

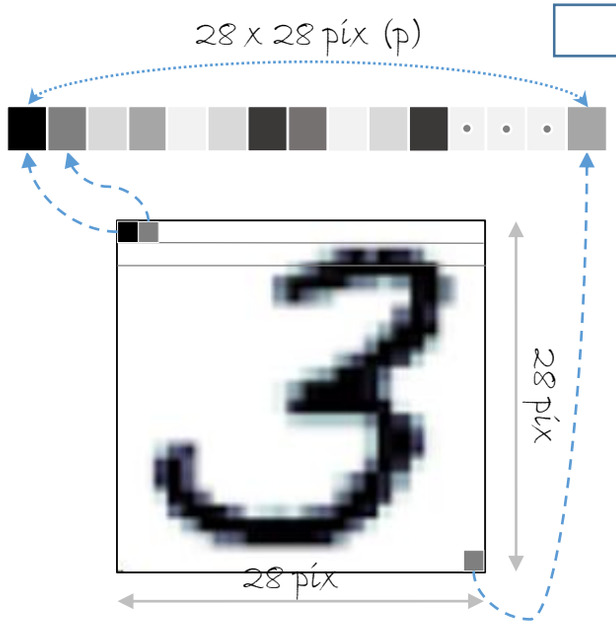
Corresponding Labels

- Data set of hand written digits with
  - ✓60,000 training images
  - ✓10,000 test images
- Each image is: 28 x 28 pixels
- Performance with different classifiers (error rate):
  - ✓Neural nets (2-layers): 1.6 %
  - ✓Deep nets (6-layers): 0.35 %
  - ✓Conv nets (different): 0.21% - 0.31%

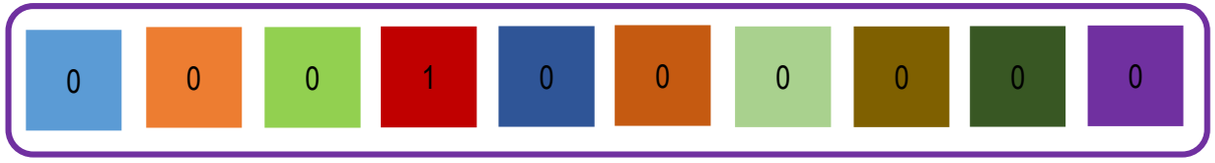
Logistic Regression



Error or Loss Function



Label One-hot encoded (Y)



Loss function

$$se = \sum_{j=0}^{9} (y_j - p_j)^2$$

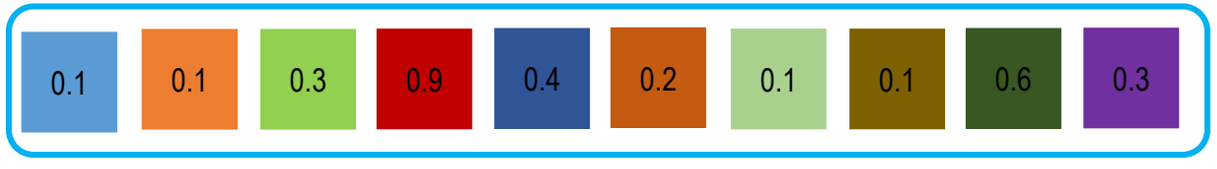
Squared error

$$ce = \sum_{j=0}^{9} y_j \, log(p_j)$$

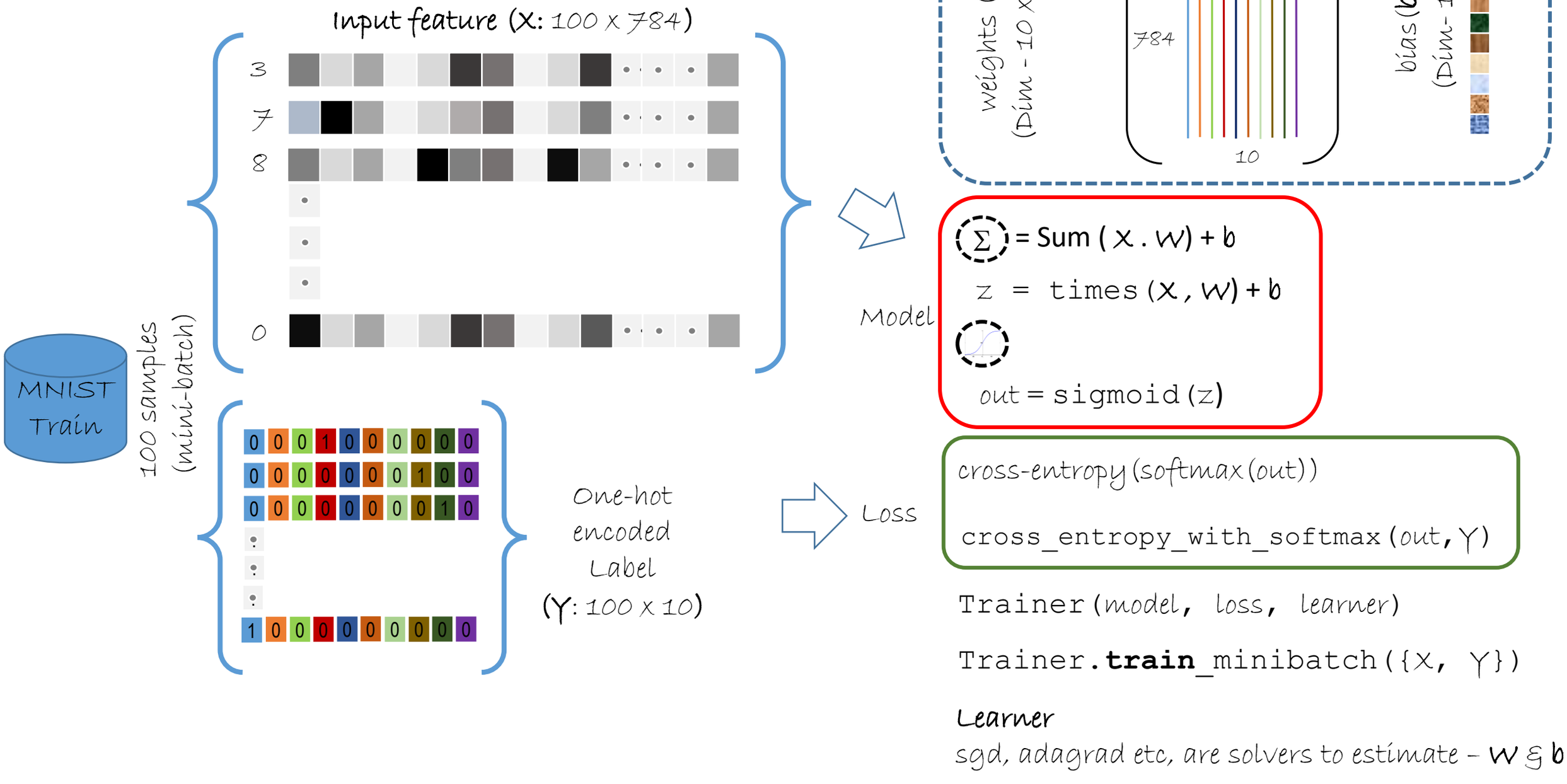
Cross entropy error

Model (w, b)

Predicted Probabilities (p)



Workflow


Input feature (X: 100 x 784)
3
7
8
0
MNIST Train
100 samples (mini-batch)
One-hot encoded Label (Y: 100 x 10)
Model
weights (W) (Dim - 10 x 784)
784
10
bias (b) (Dim- 10)
Σ = Sum (X . W) + b
z = times (X, W) + b
out = sigmoid (z)
Loss
cross-entropy (softmax (out))
cross_entropy_with_softmax (out, Y)
Trainer (model, loss, learner)
Trainer.train_minibatch ({X, Y})
Learner
sgd, adagrad etc, are solvers to estimate – W & b

"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."

"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$
$$h_2 = \sigma(W_2 h_1 + b_2)$$
$$P = \text{softmax}(W_{\text{out}} h_2 + b_{\text{out}})$$

with input $x \in \mathrm{R}^M$

"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$
$$h_2 = \sigma(W_2 h_1 + b_2)$$
$$P = \mathrm{softmax}(W_{\mathrm{out}} h_2 + b_{\mathrm{out}})$$

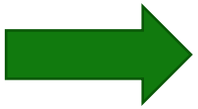
with input $x \in \mathrm{R}^M$ and one-hot label $y \in \mathrm{R}^J$
and cross-entropy training criterion

$$ce = y^{\mathrm{T}} \log P$$
$$\sum_{\mathrm{corpus}} ce = \max$$

"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."

example: 2-hidden layer feed-forward NN

$$h_1 = \sigma(W_1 x + b_1)$$
$$h_2 = \sigma(W_2 h_1 + b_2)$$
$$P = \text{softmax}(W_{\text{out}} h_2 + b_{\text{out}})$$



```
h1 = sigmoid (x  @ W1   + b1)
h2 = sigmoid (h1 @ W2   + b2)
P  = softmax (h2 @ Wout + bout)
```

with input $x \in \mathrm{R}^M$ and one-hot label $y \in \mathrm{R}^J$
and cross-entropy training criterion

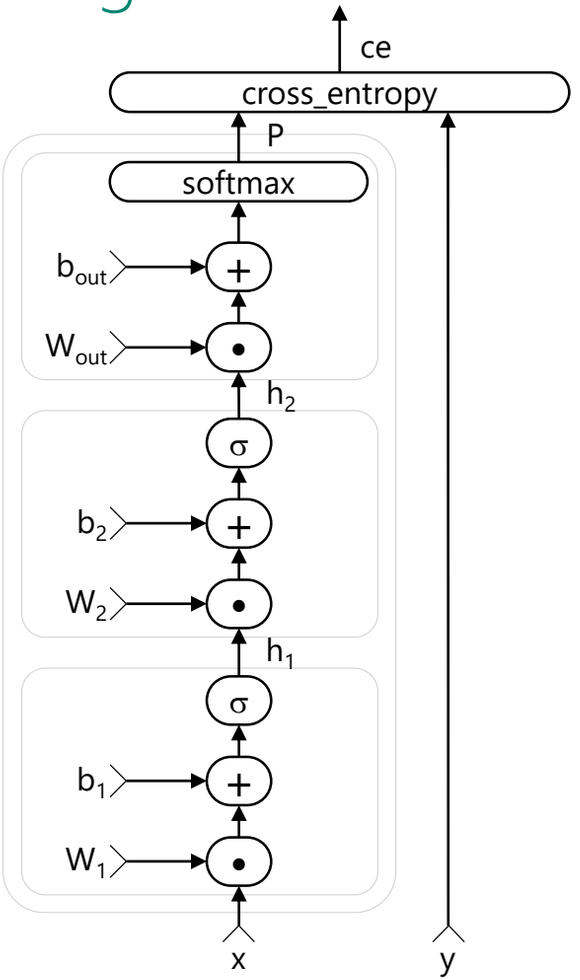
$$ce = y^{\mathrm{T}} \log P$$
$$\sum_{\text{corpus}} ce = \max$$

```
ce = cross_entropy (P, y)
```

"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."

```
h1 = sigmoid (x  @ W1   + b1)
h2 = sigmoid (h1 @ W2   + b2)
P  = softmax (h2 @ Wout + bout)
ce = cross_entropy (P, y)
```

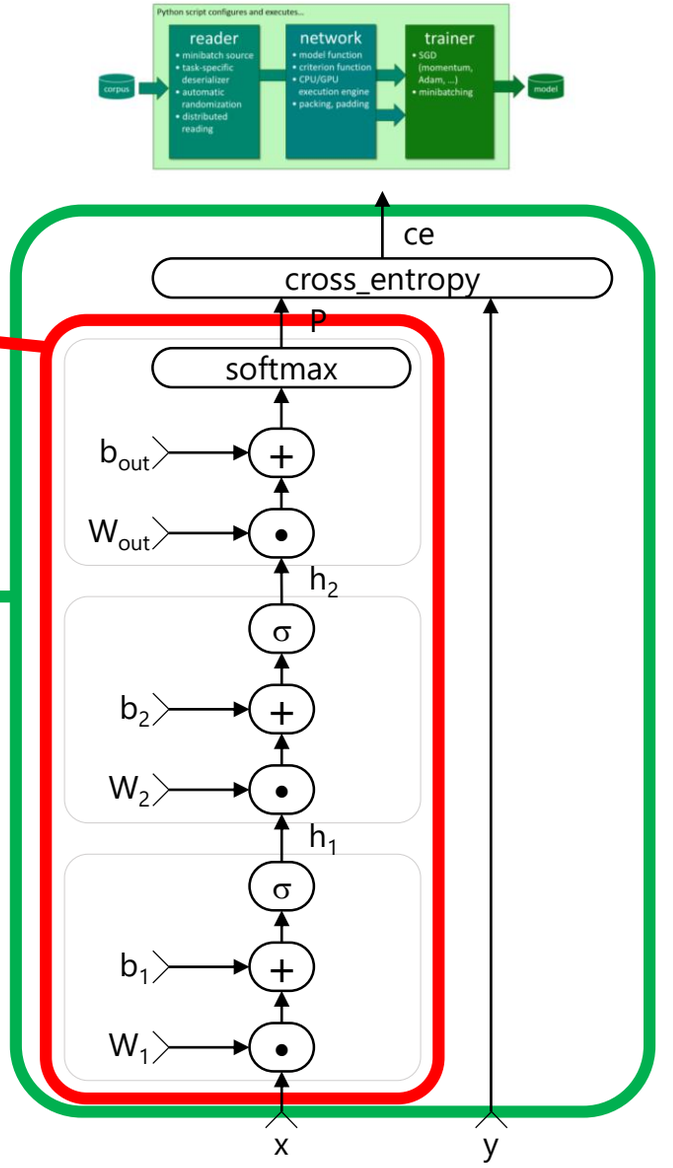
“CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications.”



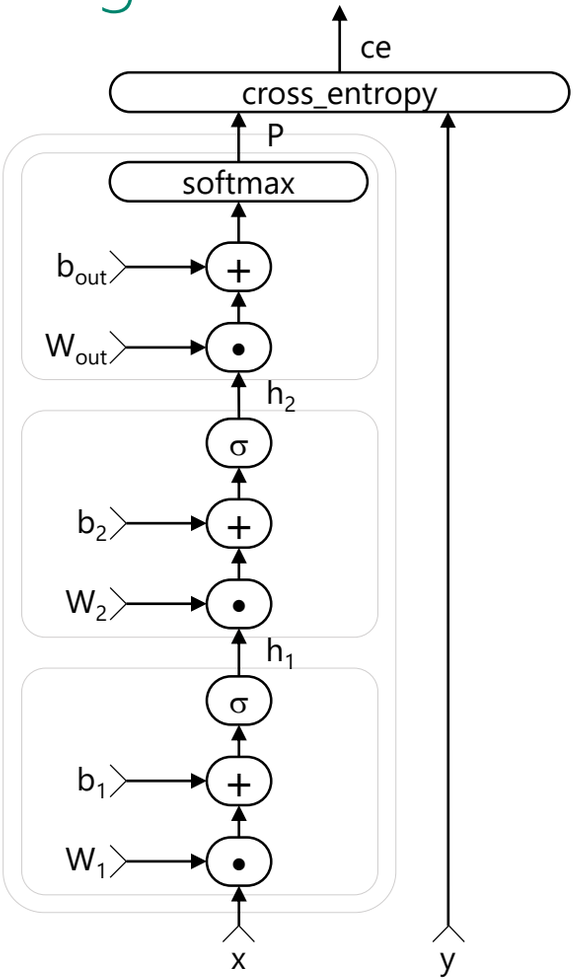

```
h1 = sigmoid (x  @ W1   + b1)
h2 = sigmoid (h1 @ W2   + b2)
P  = softmax (h2 @ Wout + bout)
ce = cross_entropy (P, y)
```

Network / Model

- the model function
  - *features* → *predictions*
  - defines the **model structure** & parameter initialization
  - holds parameters that will be learned by training
- the criterion function
  - (*features*, *labels*) → (*training loss*, *additional metrics*)
  - defines **training** and **evaluation criteria** on top of the model function
  - provides gradients w.r.t. training criteria
- both written in Python (CNTK V2 API)
  - looks like Python functions
  - hides graph business underneath



"CNTK expresses (nearly) arbitrary neural networks by composing simple building blocks into complex computational networks, supporting relevant network types and applications."



- nodes: functions (primitives)
  - can be composed into reusable composites
- edges: values
  - incl. tensors, sparse
- automatic differentiation
  - $\partial \mathcal{F} / \partial \text{in} = \partial \mathcal{F} / \partial \text{out} \cdot \partial \text{out} / \partial \text{in}$
- deferred computation → execution engine
- editable, clonable

LEGO-like composability allows CNTK to support wide range of networks & applications