Machine Learning using Python

Neda Hantehzadeh

Week 3

Office Hours

Time: Wednesdays 5-6

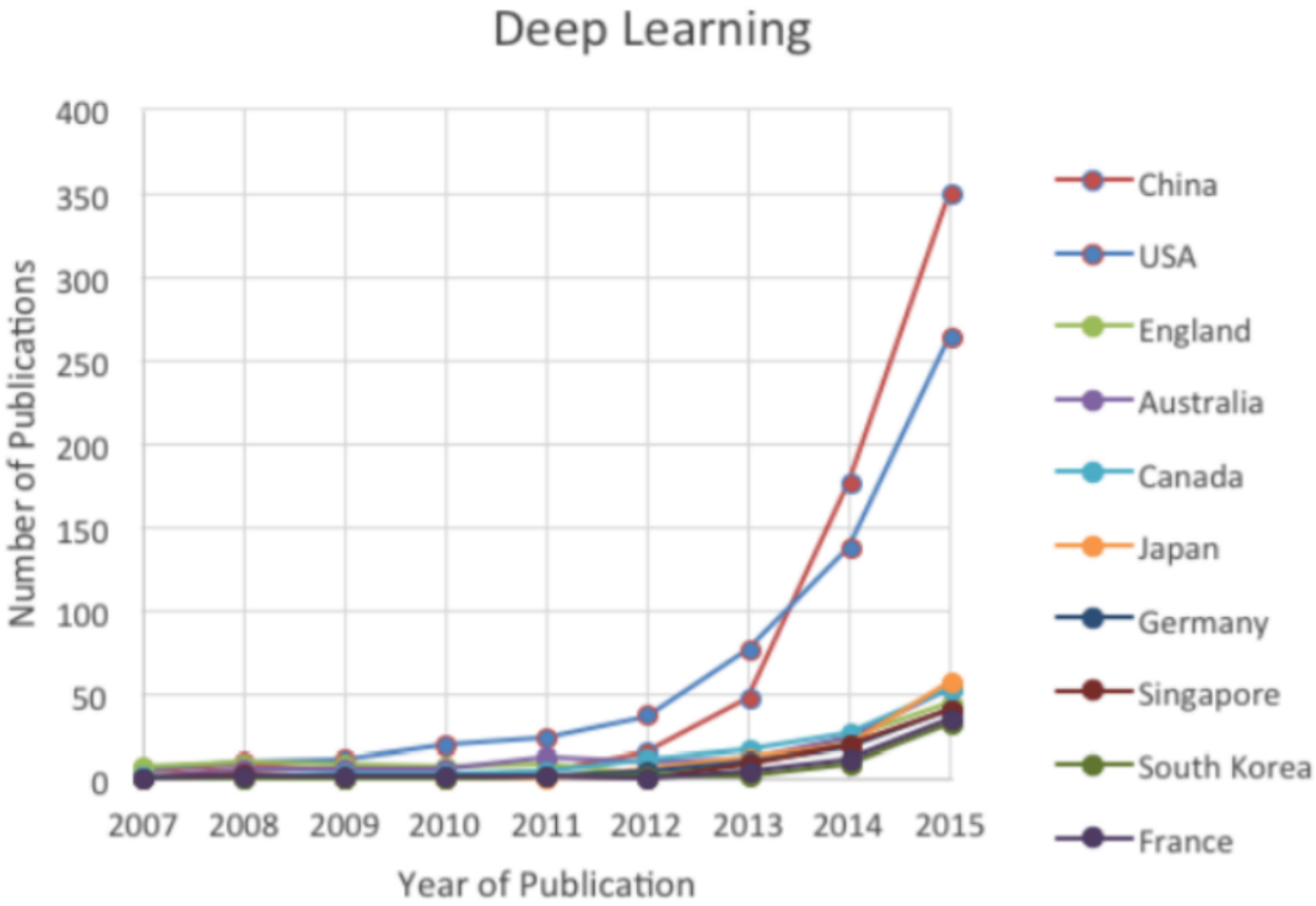
Location: IDS 594

Email: nhante2@uic.edu

Outline

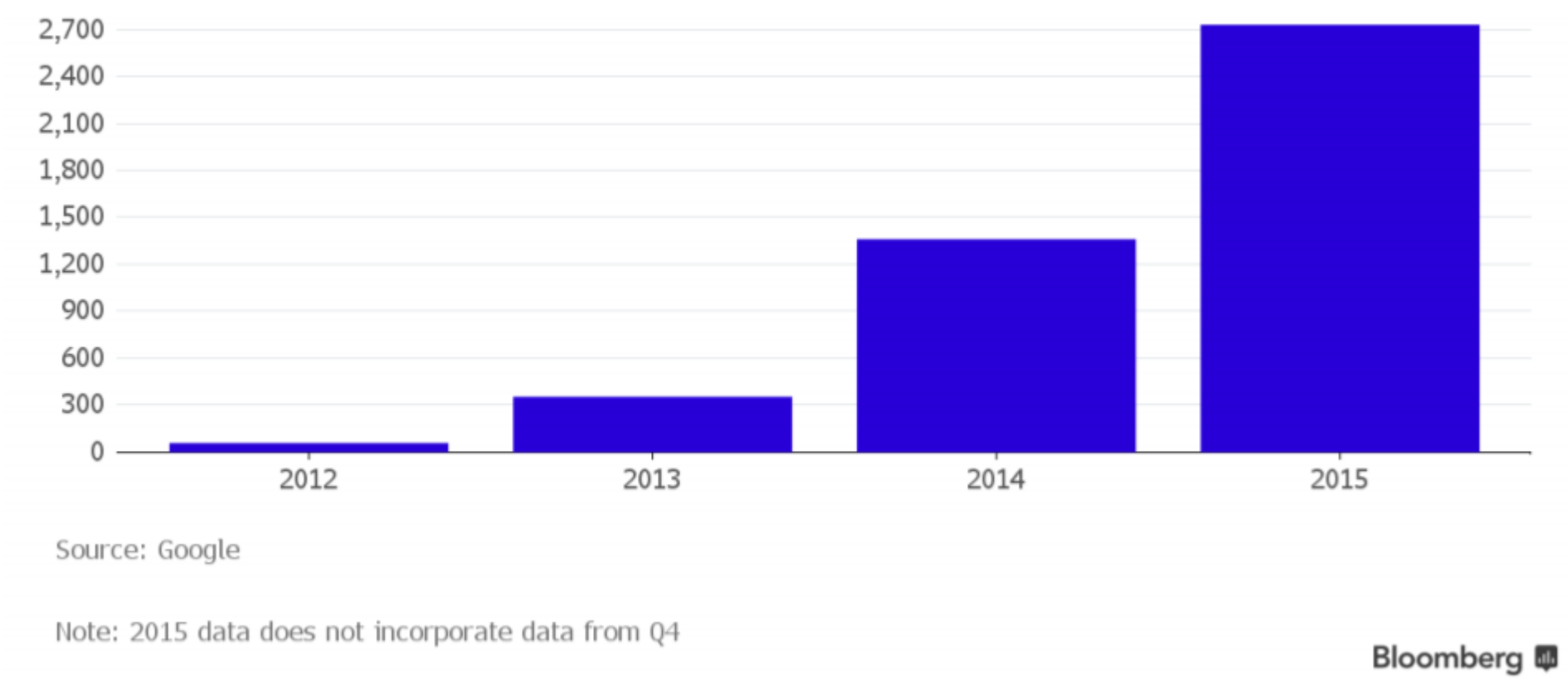
- Why is deep learning taking off?
- Deep Learning vs Traditional Machine Learning
- Type of Deep Learning Applications
- Optimization
- Learning Rate
- Back propagation

Academic Publication about Deep Learning

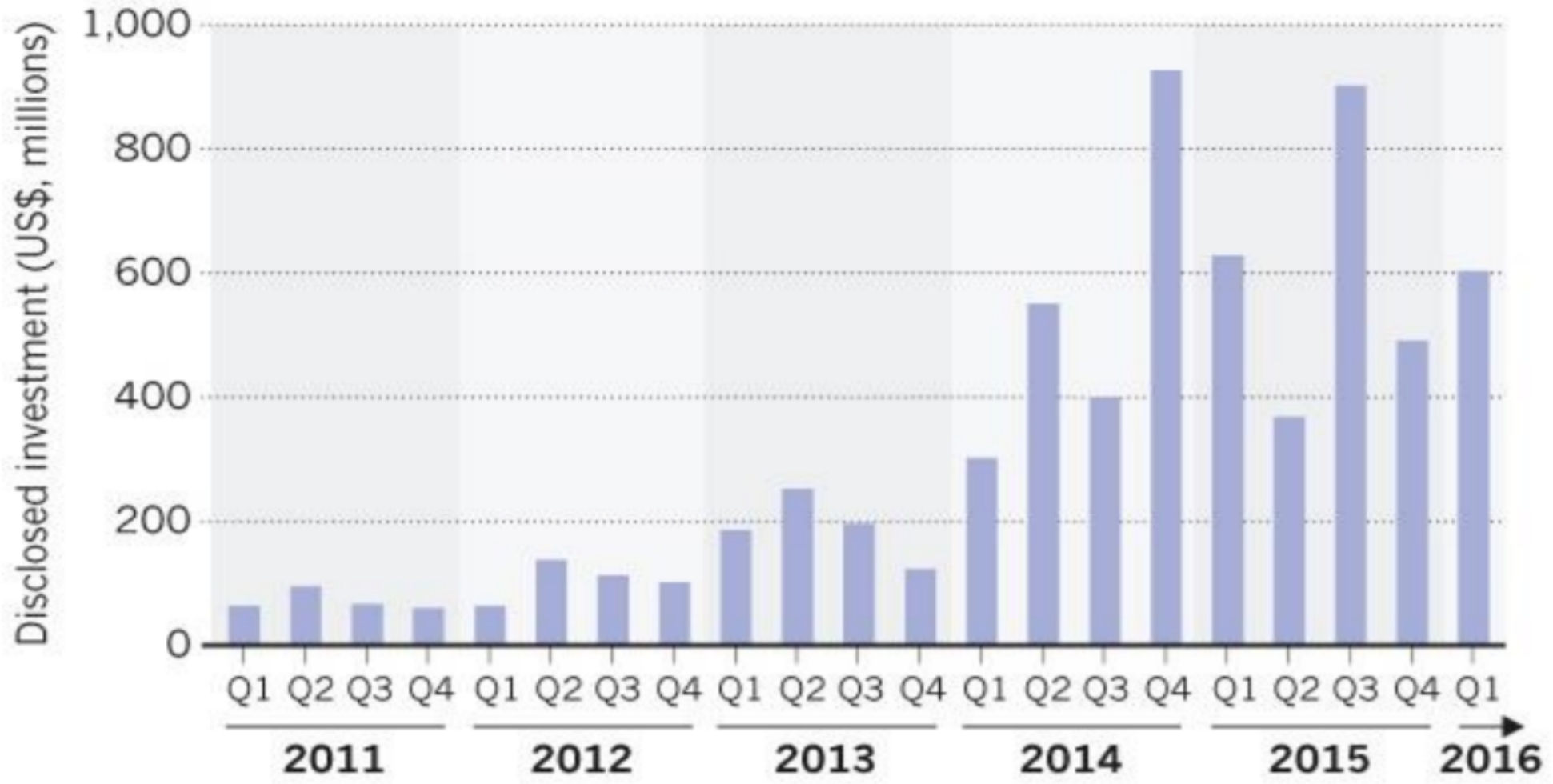


Google

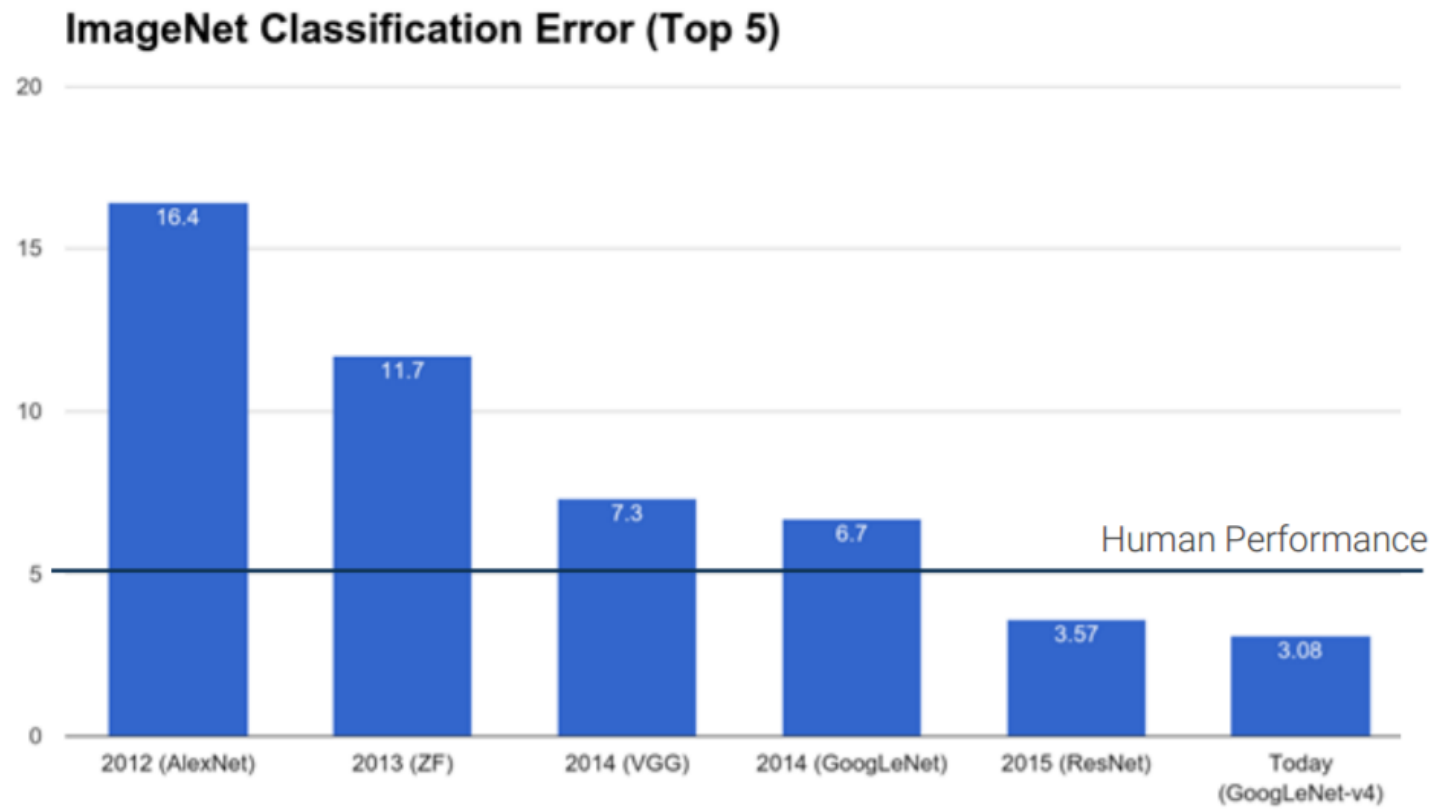
Number of software project within Google that uses a key AI technology, called Deep Learning.



Investment in technologies that uses AI has increased in recent years.

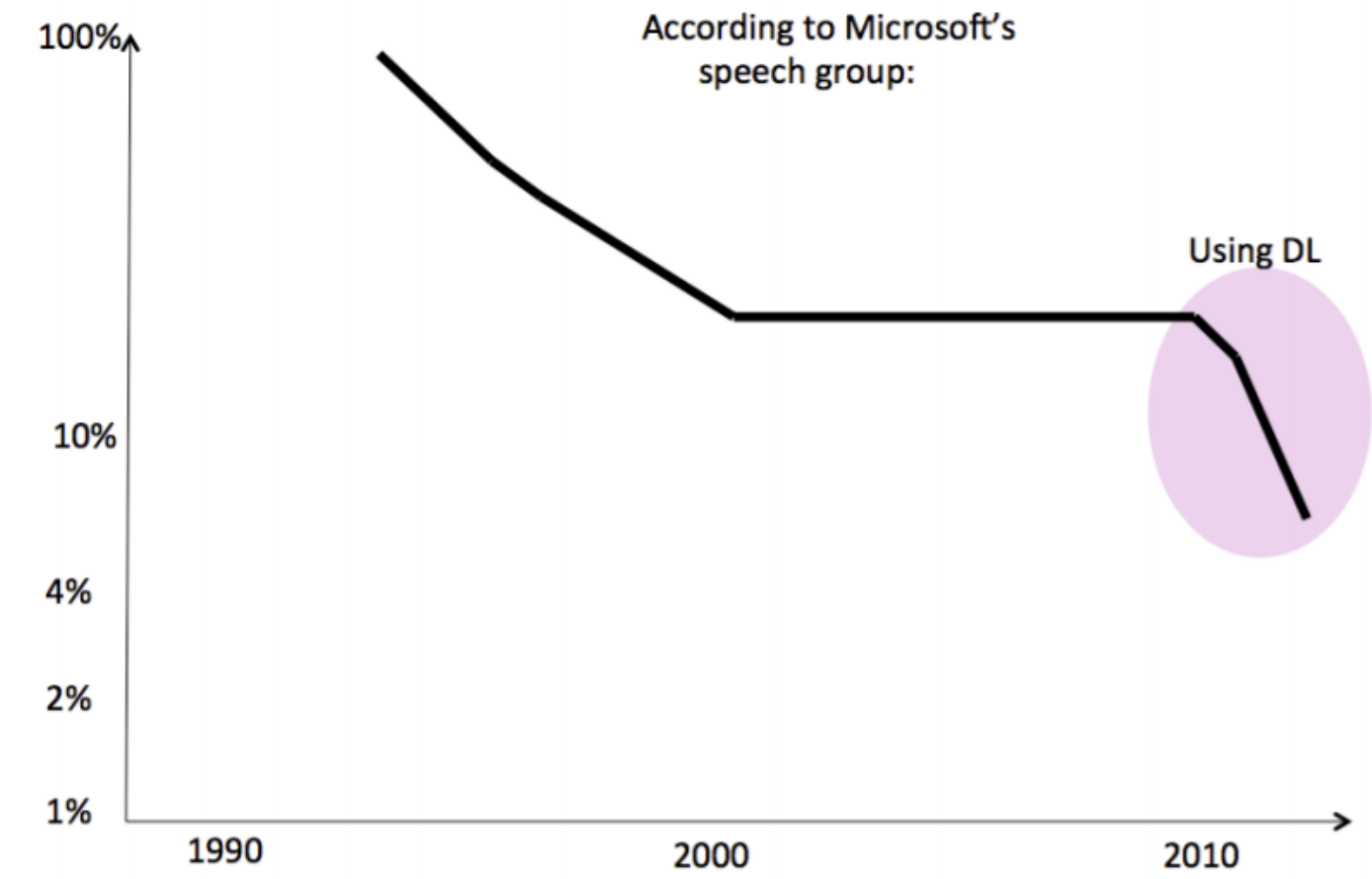


Is Deep Learning better than humans?



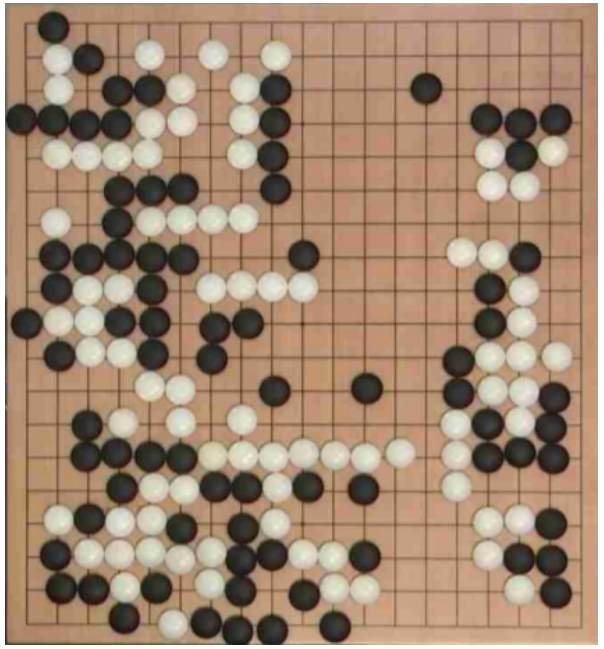
ImageNet: The “computer vision World Cup”

Is Deep Learning better than humans?



Deep Learning in Speech Recognition

WHY DEEP LEARNING IS EXCITING?



Google’s AlphaGo wins the match against Go champion Lee Sedol

WHY DEEP LEARNING IS EXCITING?

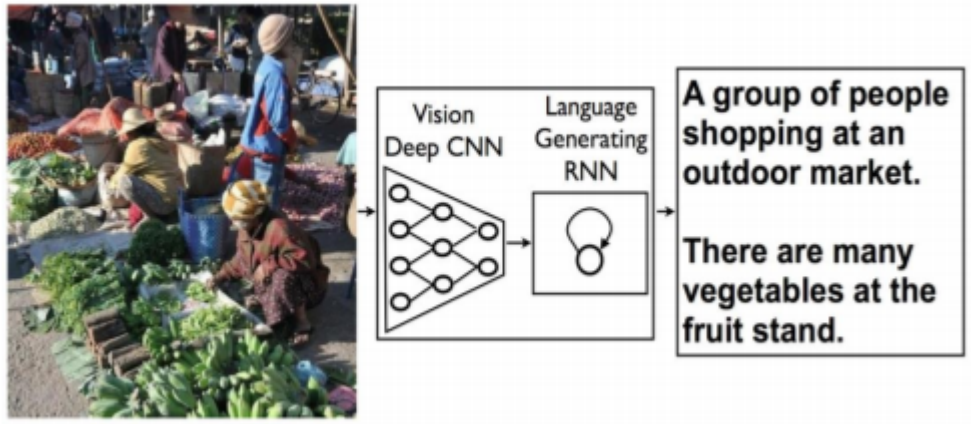
Google’s self driving cars has accomplished
126,000 miles of driving autonomously

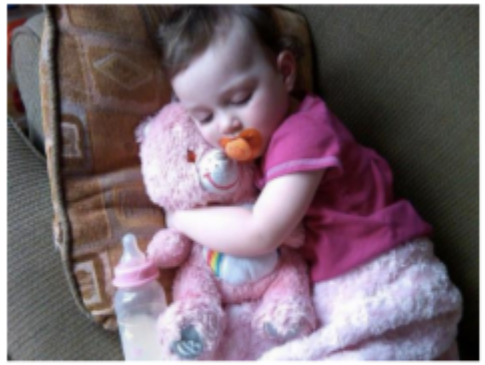
Ongoing Research with Deep Learning

Image Segmentation

Ongoing Research with Deep Learning

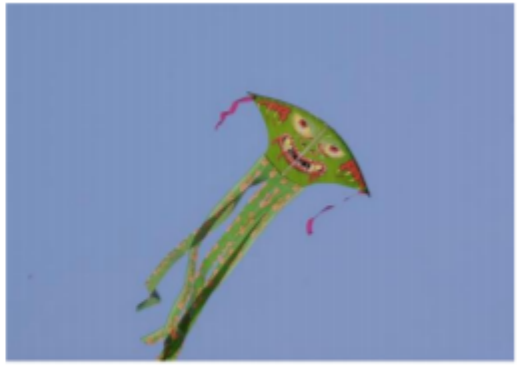
Image Captioning





A close up of a child holding a stuffed animal

Two pizzas sitting on top of a stove top oven



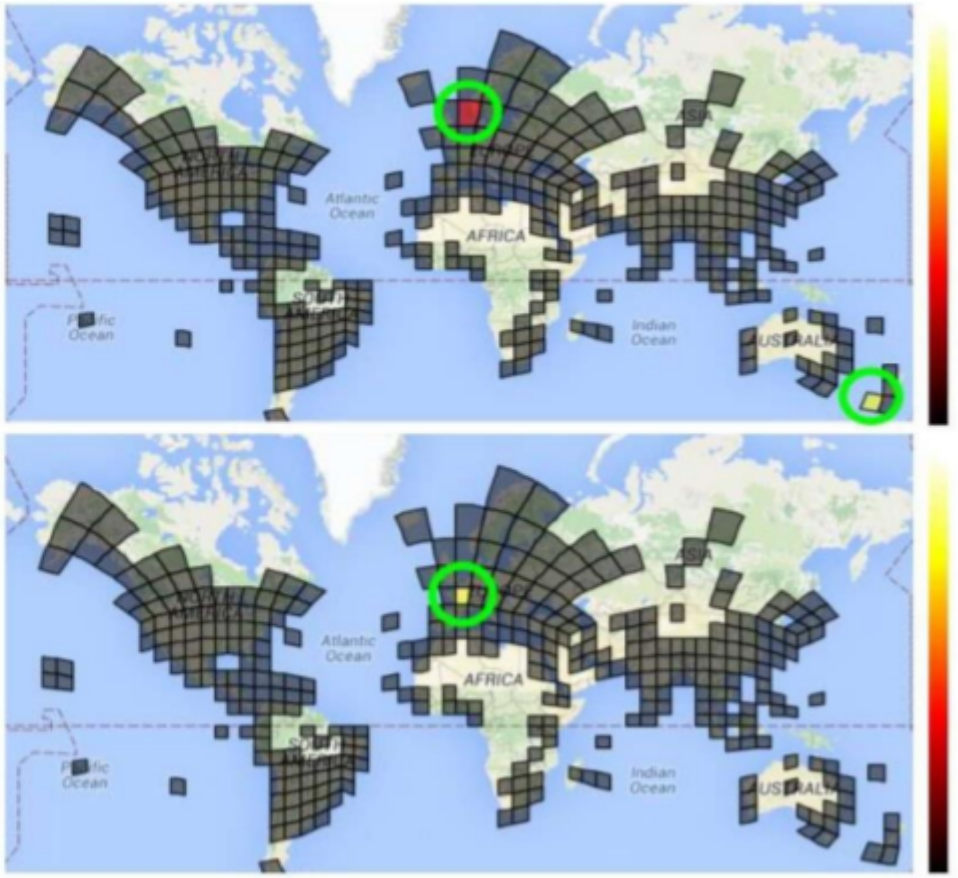
A man flying through the air while riding a skateboard

Ongoing Research with Deep Learning

Image Localization

Photo CC-BY-NC by edwin.11

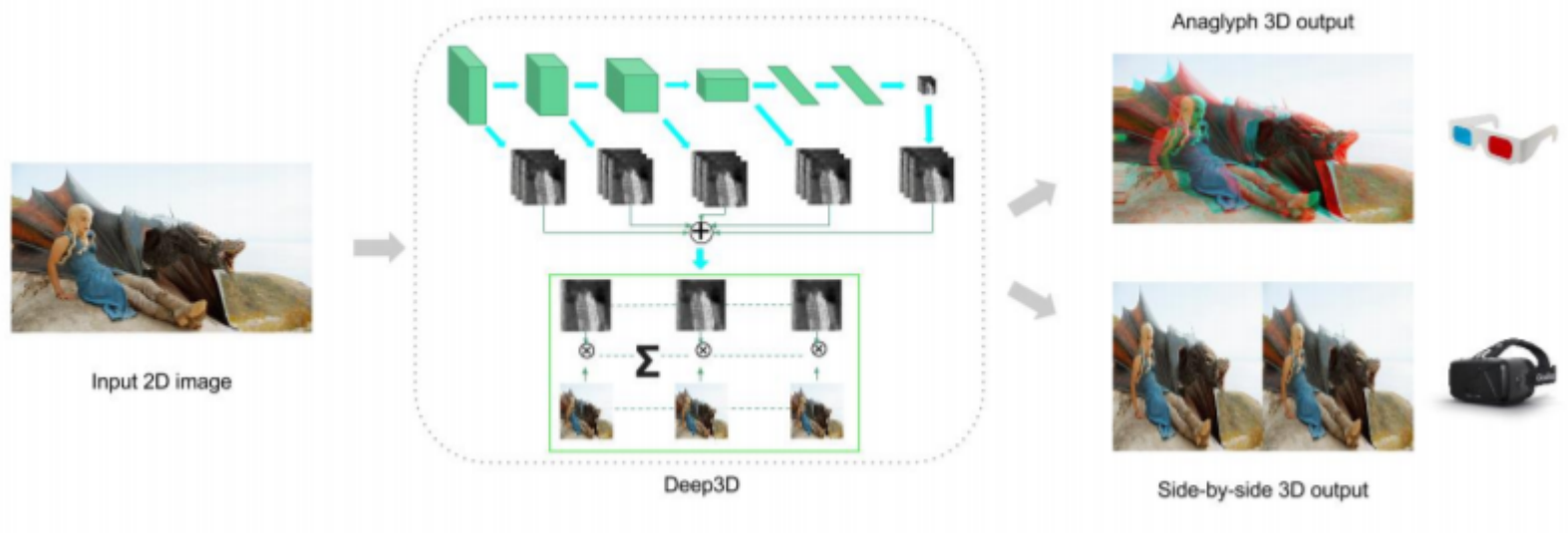
Photo CC-BY-NC by stevekc



PlaNet is able to determine the location of almost any image…

Ongoing Research with Deep Learning

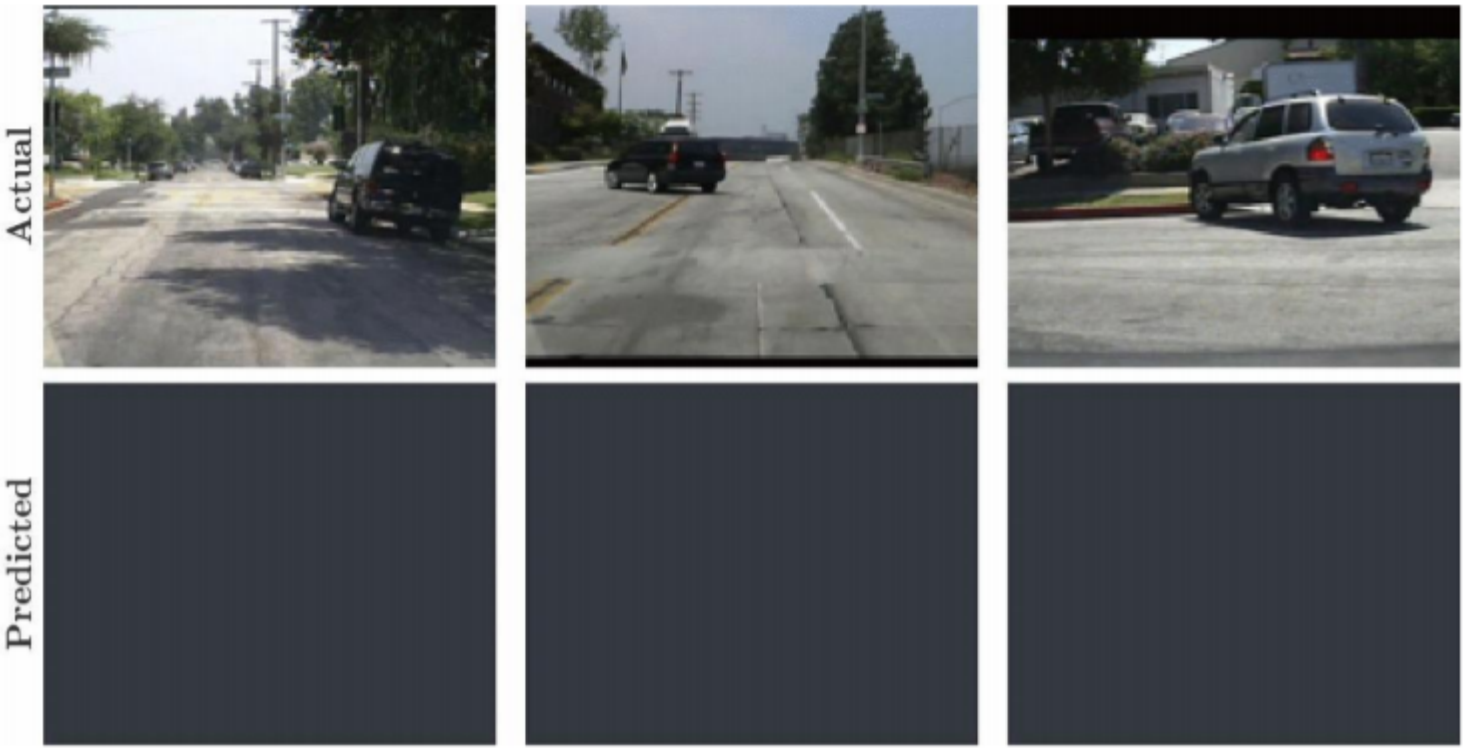
Image Transformation-2D to 3D



Deep 3D can automatically convert image/video from 2D to 3D with Convolutional Neural Networks. It learns to infer 3D representations of the world based on training set of 3D movies.

Ongoing Research with Deep Learning

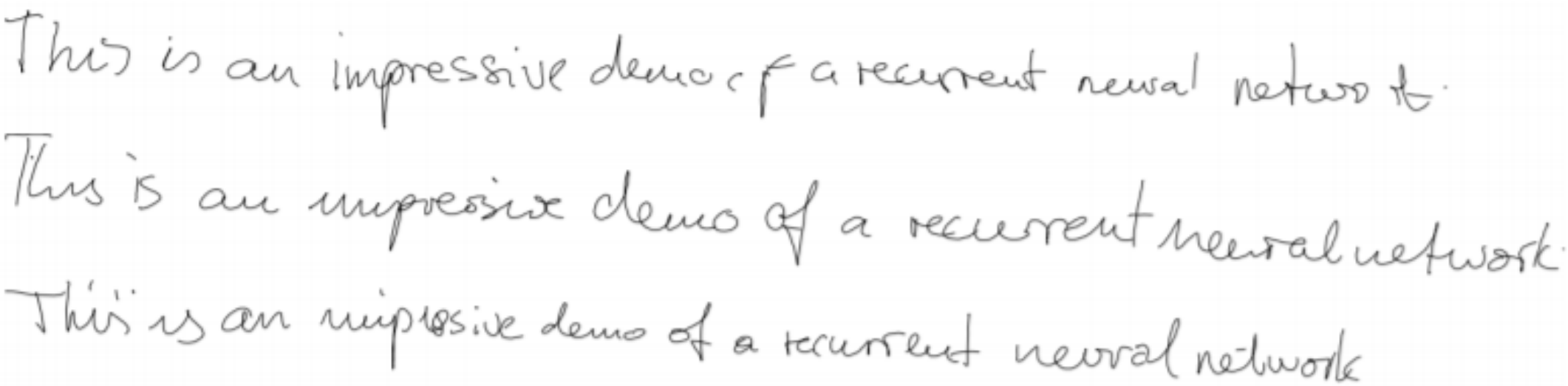
Video Sequence Prediction



PredNet- a deep convolutional recurrent neural network that predicts the future frames in a video sequence. These networks are able to robustly learn to predict the movement of synthetic (rendered) objects.

Ongoing Research with Deep Learning

Handwriting generation



This LSTM recurrent neural network is able to generate highly realistic cursive handwriting in a wide variety of styles.

DeepDream- Inceptionism

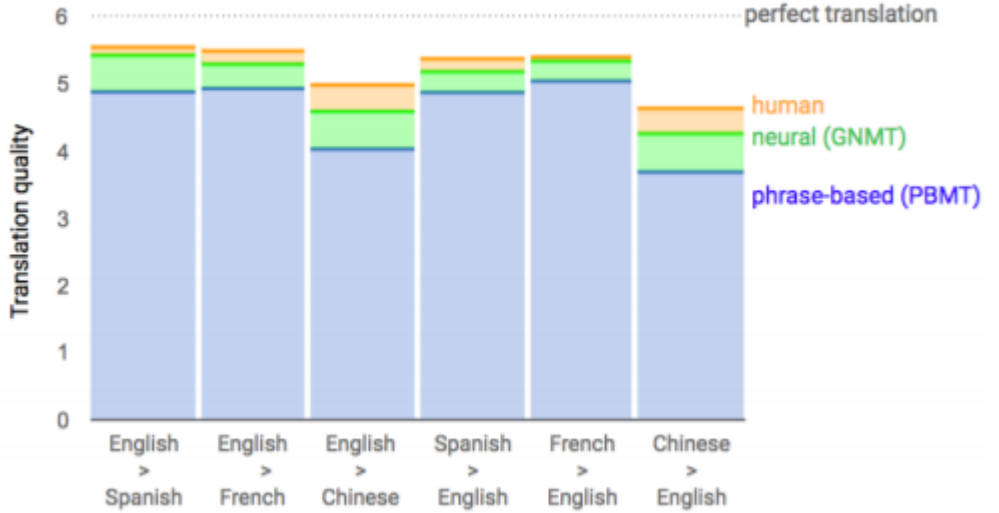
A tool for Artists?

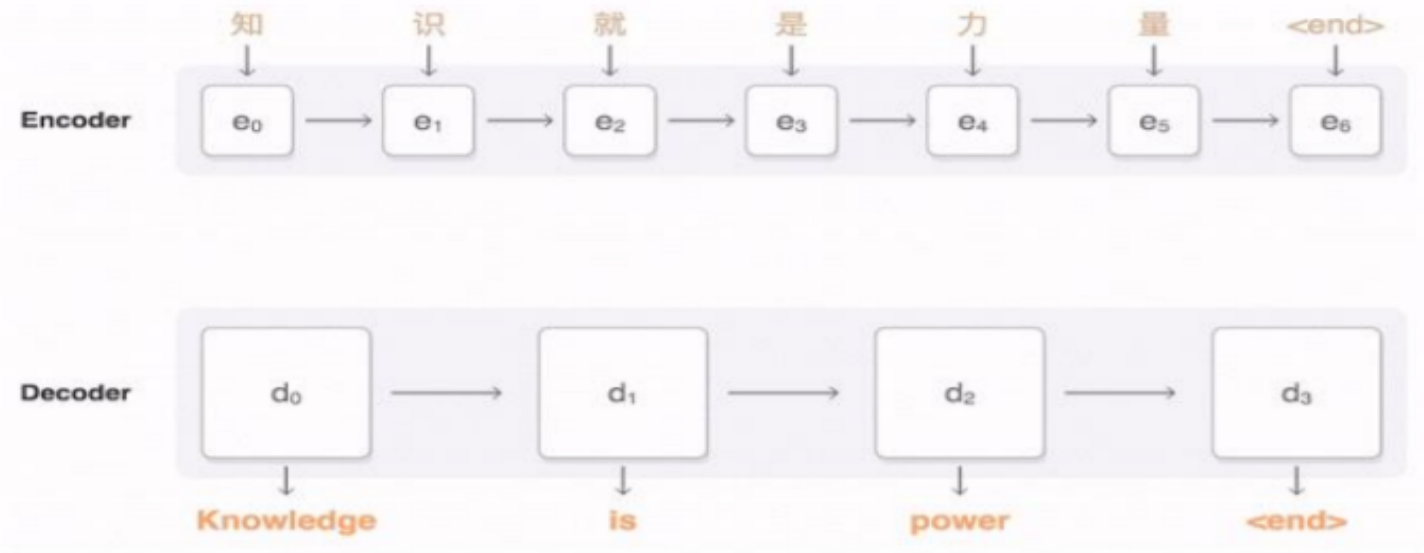
The tool was originally build for scientist and engineers to see what a deep neural network is seeing when it is looking in a given image. Later the algorithm became a new form of abstract art.

Deep Learning in NLP

Translation

Google Neural Machine translation uses LSTM architecture and reduces translation error by more than 55%-85% compared to traditional approaches.





Deep Learning in NLP

Neural Chatbot: Predict the next sentence given the previous sentences in a conversation

Human: what is the purpose of life ?

Machine: to serve the greater good .

Human: what is the purpose of living ?

Machine: to live forever

Human: what is the purpose of existence?

Machine: to find out what happens when we get to the planet earth

Human: where are you now ?

Machine: i 'm in the middle of nowhere .

Human: what is the purpose of dying ?

Machine: to have a life

Human: what is the purpose of being intelligent ?

Machine: to find out what it is.

Human: what is the purpose of emotions ?

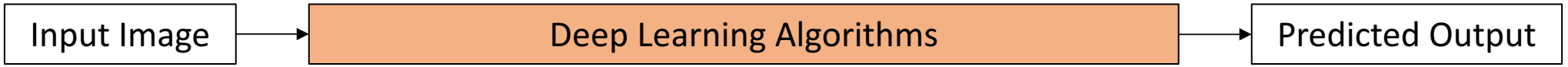
Machine: I don 't know .

Philosophical Q&A – Network trained with OpenSubtitles (movie subtitles)

Is Deep Learning another term for AI?

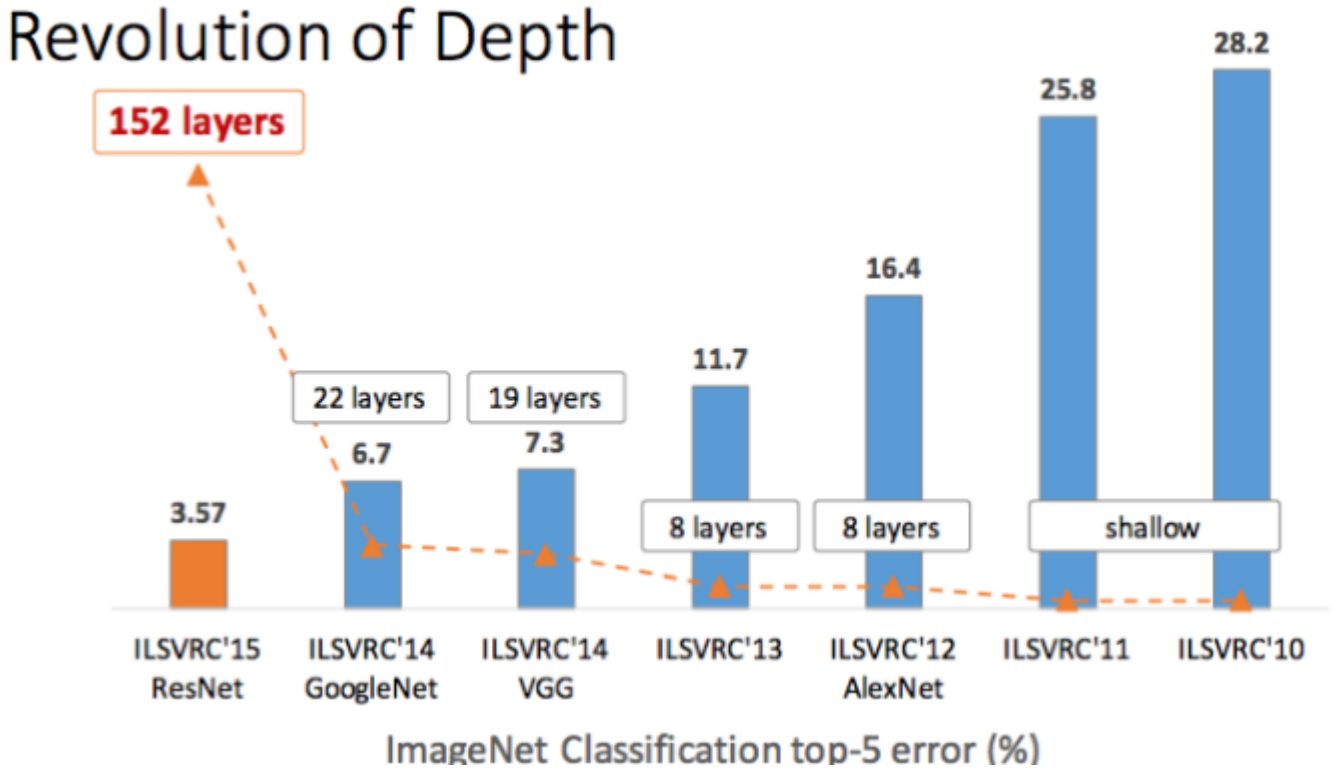
Yes but No. Deep learning is part of AI and the flow is different compare to traditional AI linear programming structures.

Traditional Machine Learning Flow



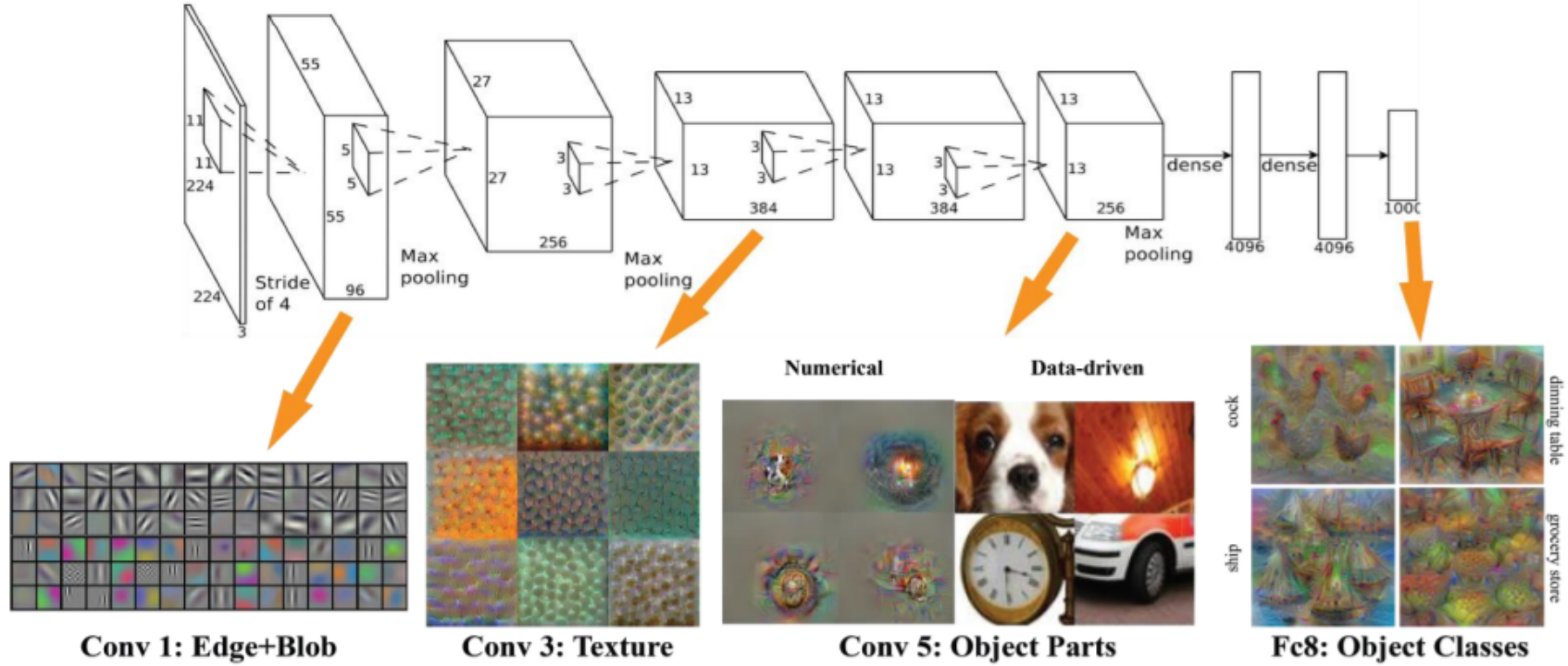
Deep Learning Flow

Does more layers mean better performance?



More layers the network has, the higher level features it will learn.

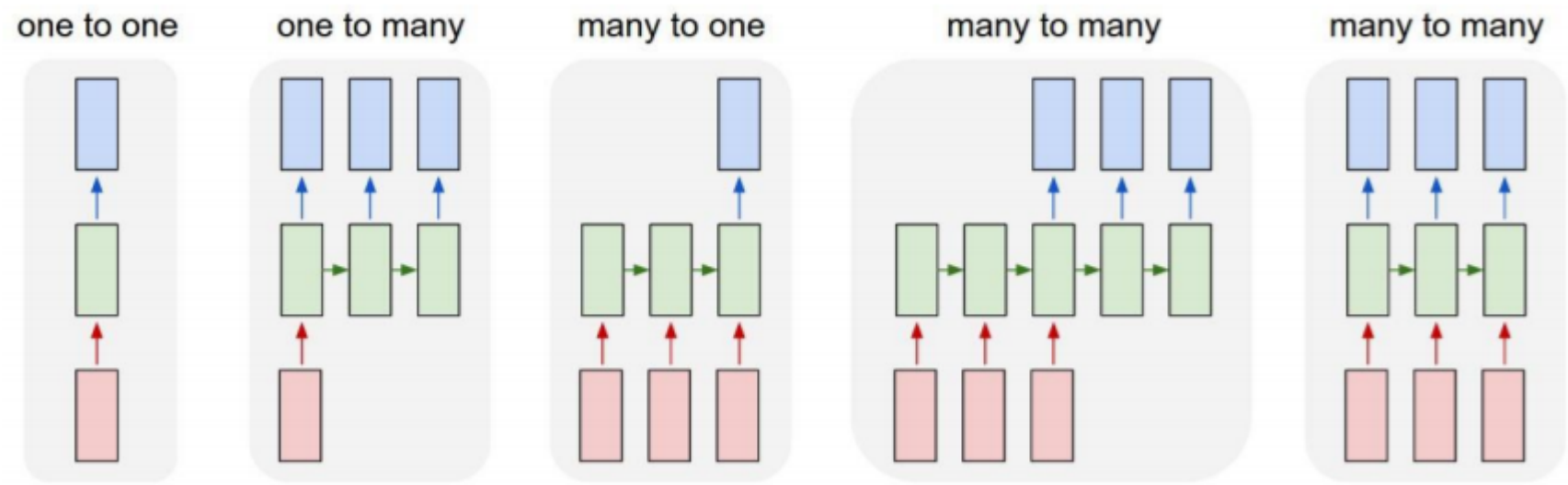
Deep Learning- Convolutional Neural Networks (CNNs)



Convoltion layer will filter out features and will automatically learn from input using a convolution kernel.

Pooling layer will compute the average or maximum of a feature over a region and will downsize the input. It also helps to detect objects in some unusual places and will reduce memory.

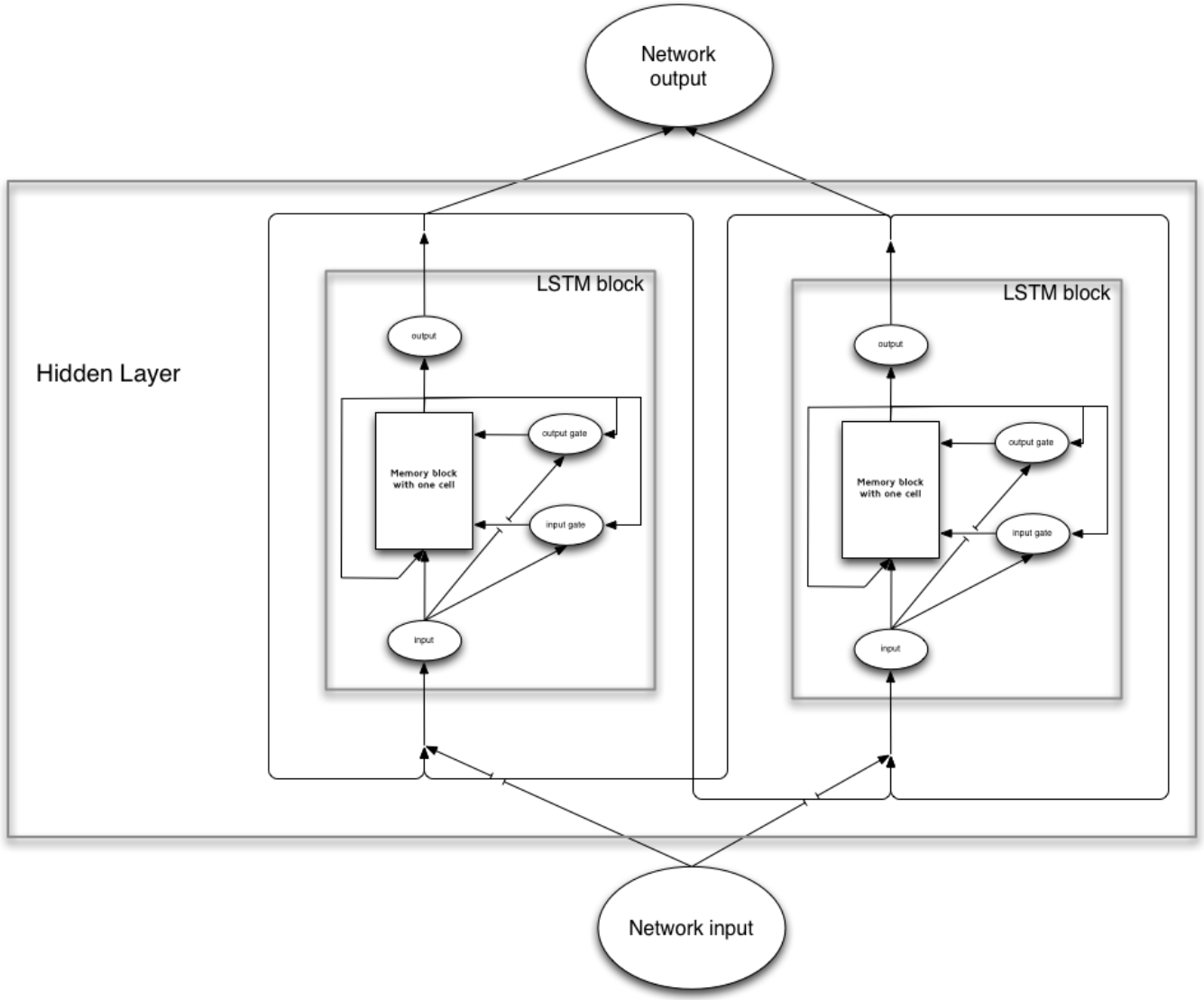
Deep Learning- Recurrent Neural Nets (RNN)



RNNs are general computers which can learn algorithms to map input sequences to output sequences (flexible-sized vectors). The output vector's contents are influenced by the entire history of inputs.
Some of the important applications are in NLP, Market prediction and handwriting prediction….

DL Architectures: LSTMs

- A LSTM network is an artificial neural network that contains LSTM blocks instead of, or in addition to, regular network units.
- A LSTM block may be described as a "smart" network unit that can remember a value for an arbitrary length of time.
- A LSTM block contains gates that determine when the input is significant enough to remember, when it should continue to remember or forget the value, and when it should output the value.
- As of 2016, major technology companies including Google, Apple, Microsoft, and Baidu are using LSTM networks as fundamental components in new products.



Recap

- We have a score function: $s = f(x; W) \overset{\text{e.g.}}{=} Wx$
- We have a loss function

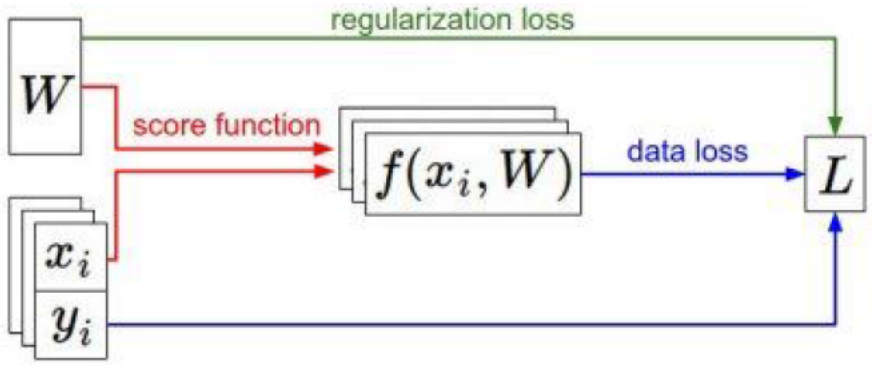
Softmax

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

SVM

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ Full loss



Q: How do we calculate the best loss?

Strategy # 1: Random Search

```
# assume X_train is the data where each column is an example (e.g. 3073 x 50,000)
# assume Y_train are the labels (e.g. 1D array of 50,000)
# assume the function L evaluates the loss function

bestloss = float("inf") # Python assigns the highest possible float value
for num in xrange(1000):
  W = np.random.randn(10, 3073) * 0.0001 # generate random parameters
  loss = L(X_train, Y_train, W) # get the loss over the entire training set
  if loss < bestloss: # keep track of the best solution
    bestloss = loss
    bestW = W
  print 'in attempt %d the loss was %f, best %f' % (num, loss, bestloss)

# prints:
# in attempt 0 the loss was 9.401632, best 9.401632
# in attempt 1 the loss was 8.959668, best 8.959668
# in attempt 2 the loss was 9.044034, best 8.959668
# in attempt 3 the loss was 9.278948, best 8.959668
# in attempt 4 the loss was 8.857370, best 8.857370
# in attempt 5 the loss was 8.943151, best 8.857370
# in attempt 6 the loss was 8.605604, best 8.605604
# ... (trunctated: continues for 1000 lines)
```

Lets test strategy #1

```
# Assume X_test is [3073 x 10000], Y_test [10000 x 1]
scores = Wbest.dot(Xte_cols) # 10 x 10000, the class scores for all test examples
# find the index with max score in each column (the predicted class)
Yte_predict = np.argmax(scores, axis = 0)
# and calculate accuracy (fraction of predictions that are correct)
np.mean(Yte_predict == Yte)
# returns 0.1555
```

30% accuracy! Is this good?

Strategy # 2: Calculate the slope

In one-dimension, the derivative of a function is:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

In multiple-dimension, the derivative of a function is a vector of partial derivatives…

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (first dim)**:**

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25322

gradient dW:

[?,
?,
?,
?,
?,
?,
?,
?,
?,…]

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (first dim):

[0.34 + **0.0001**,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25322

gradient dW:

[**-2.5**,
?,
?,

(1.25322 - 1.25347)/0.0001
= -2.5

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,
?,…]

current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25347

W + h (first dim):

W + h (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]

loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

Evaluate the gradient numerically

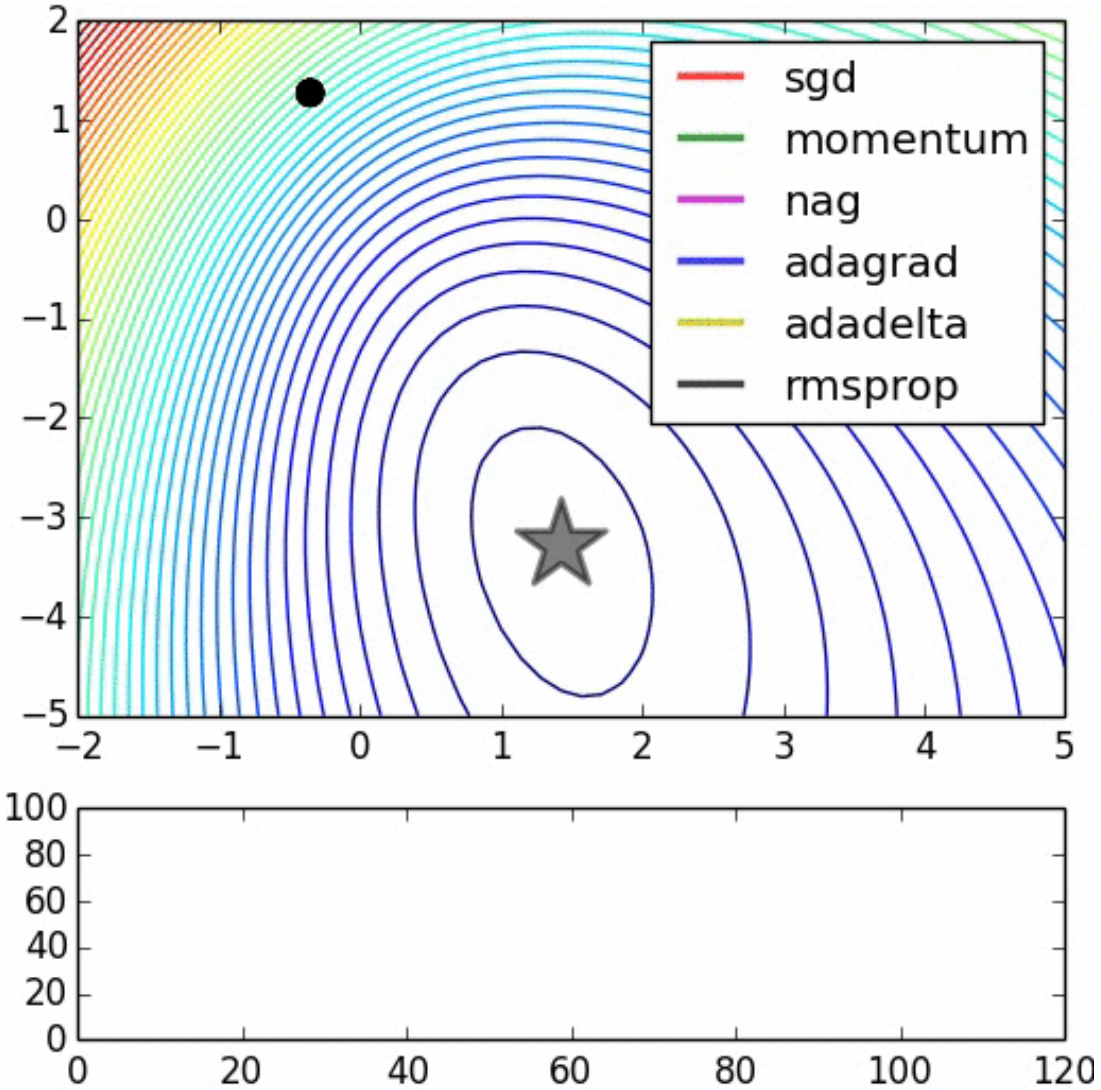
$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- Very slow
- Only approximates

```python
def eval_numerical_gradient(f, x):
  """
  a naive implementation of numerical gradient of f at x
  - f should be a function that takes a single argument
  - x is the point (numpy array) to evaluate the gradient at
  """

  fx = f(x) # evaluate function value at original point
  grad = np.zeros(x.shape)
  h = 0.00001

  # iterate over all indexes in x
  it = np.nditer(x, flags=['multi_index'], op_flags=['readwrite'])
  while not it.finished:

    # evaluate function at x+h
    ix = it.multi_index
    old_value = x[ix]
    x[ix] = old_value + h # increment by h
    fxh = f(x) # evalute f(x + h)
    x[ix] = old_value # restore to previous value (very important!)

    # compute the partial derivative
    grad[ix] = (fxh - fx) / h # the slope
    it.iternext() # step to next dimension

  return grad
```

Optimization using analytical gradient Methods

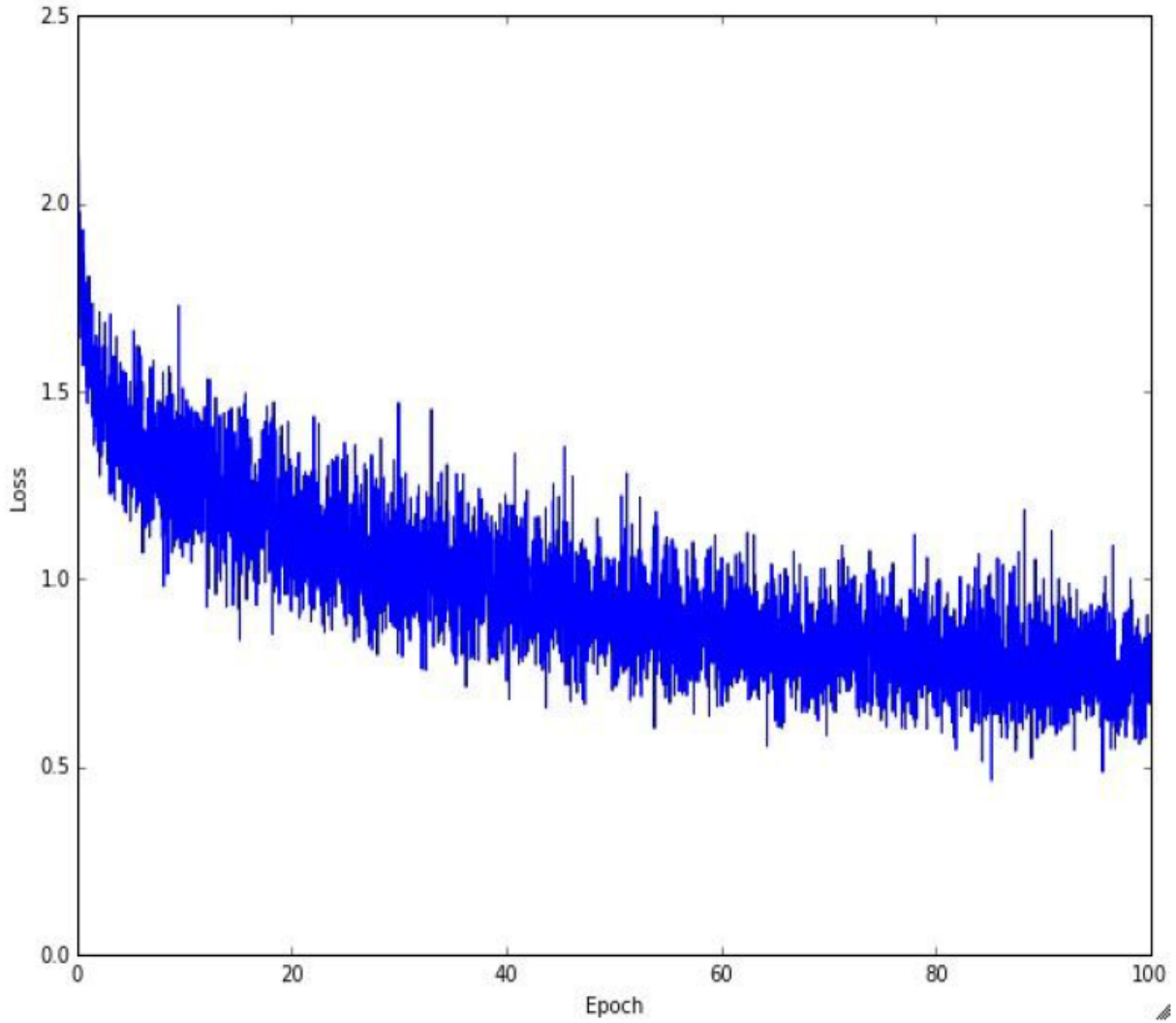
- Faster than numerical gradient
- More accurate
- Could be performed on smaller portion of train data and still be accurate



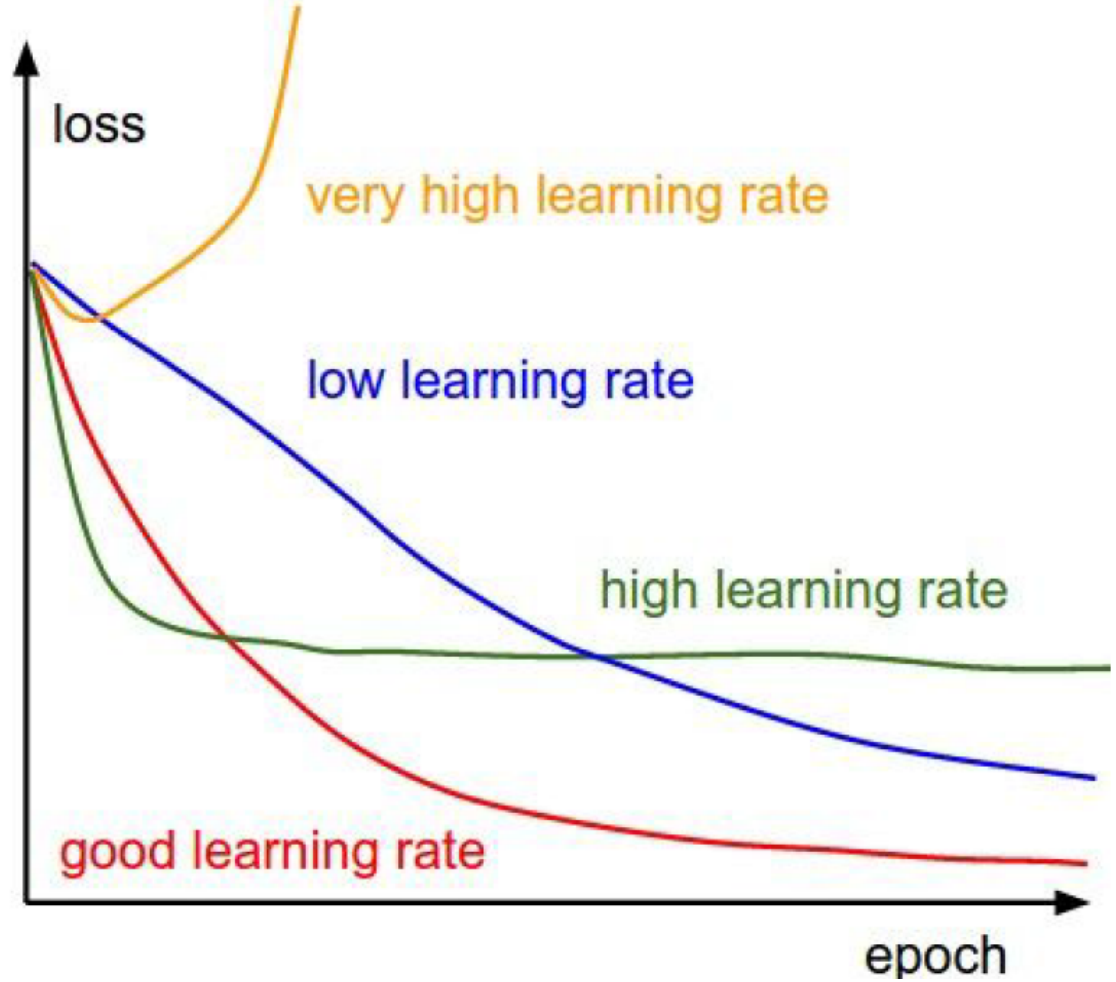
Mini-Batch Gradient Descent

```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Common mini-batch sizes are 32/64/128/256 examples



The effects of step size (or “learning rate”)



Mini-Batch Gradient Descent

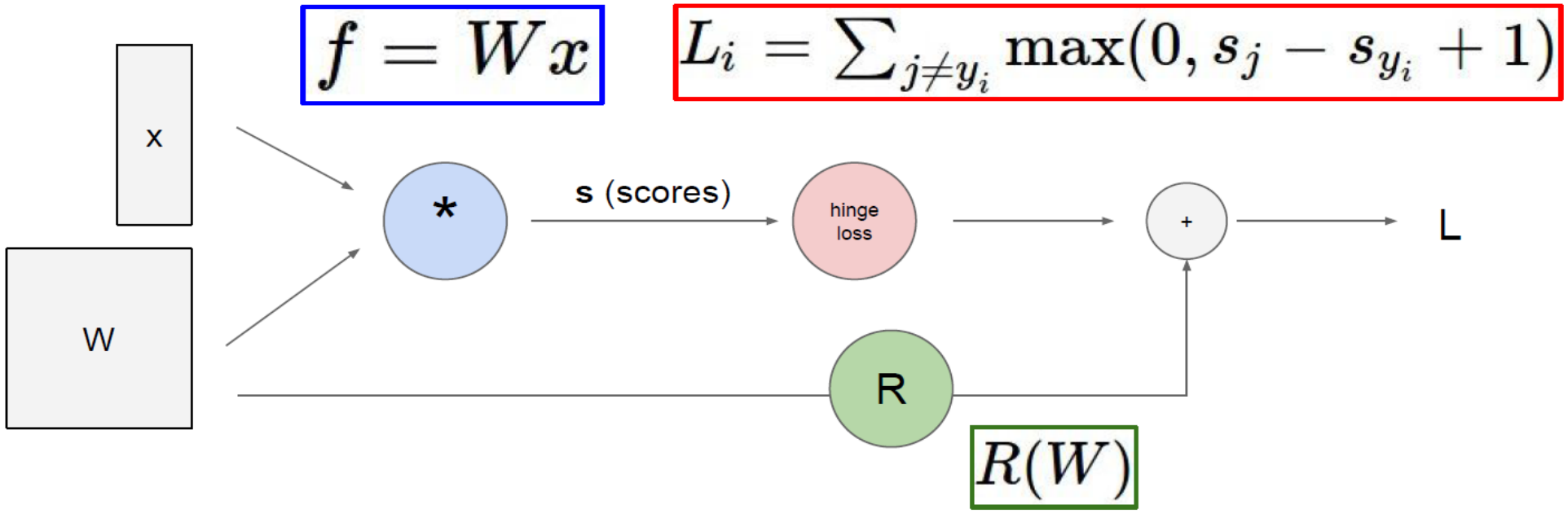
```
# Vanilla Minibatch Gradient Descent

while True:
  data_batch = sample_training_data(data, 256) # sample 256 examples
  weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Common mini-batch sizes are 32/64/128/256 examples

we will look at more fancy update formulas (momentum, Adagrad, RMSProp, Adam, …) in the code section

Backpropagation

Computational graph

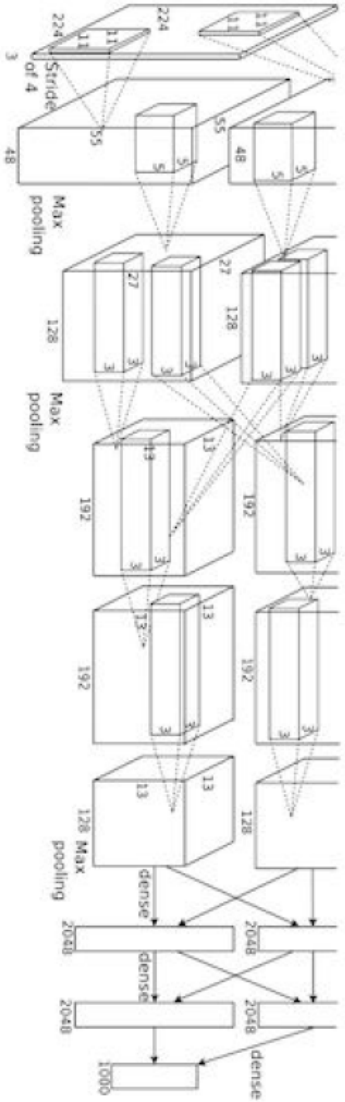


Convolutional Network (AlexNet)

input image

weights

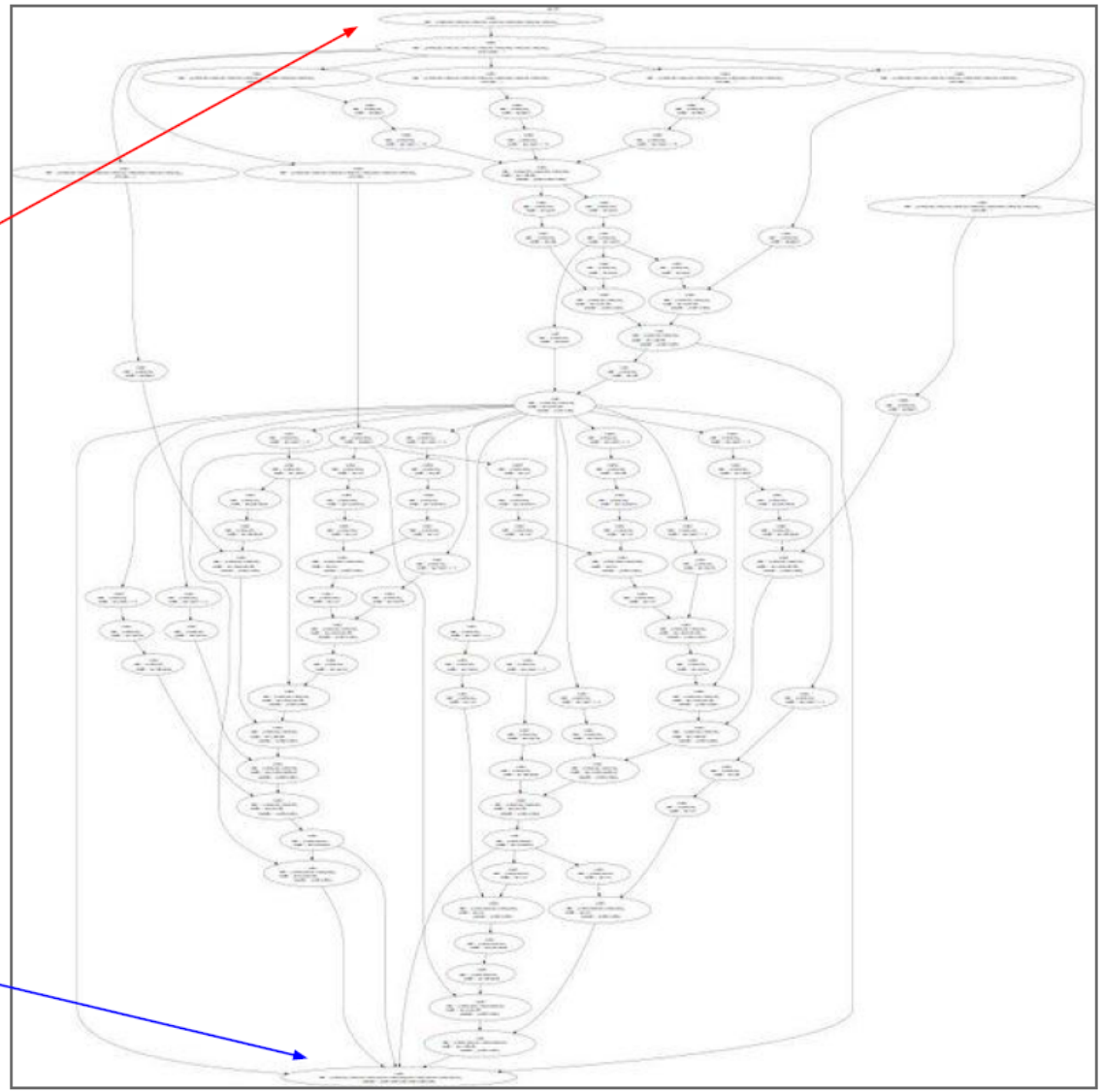
loss



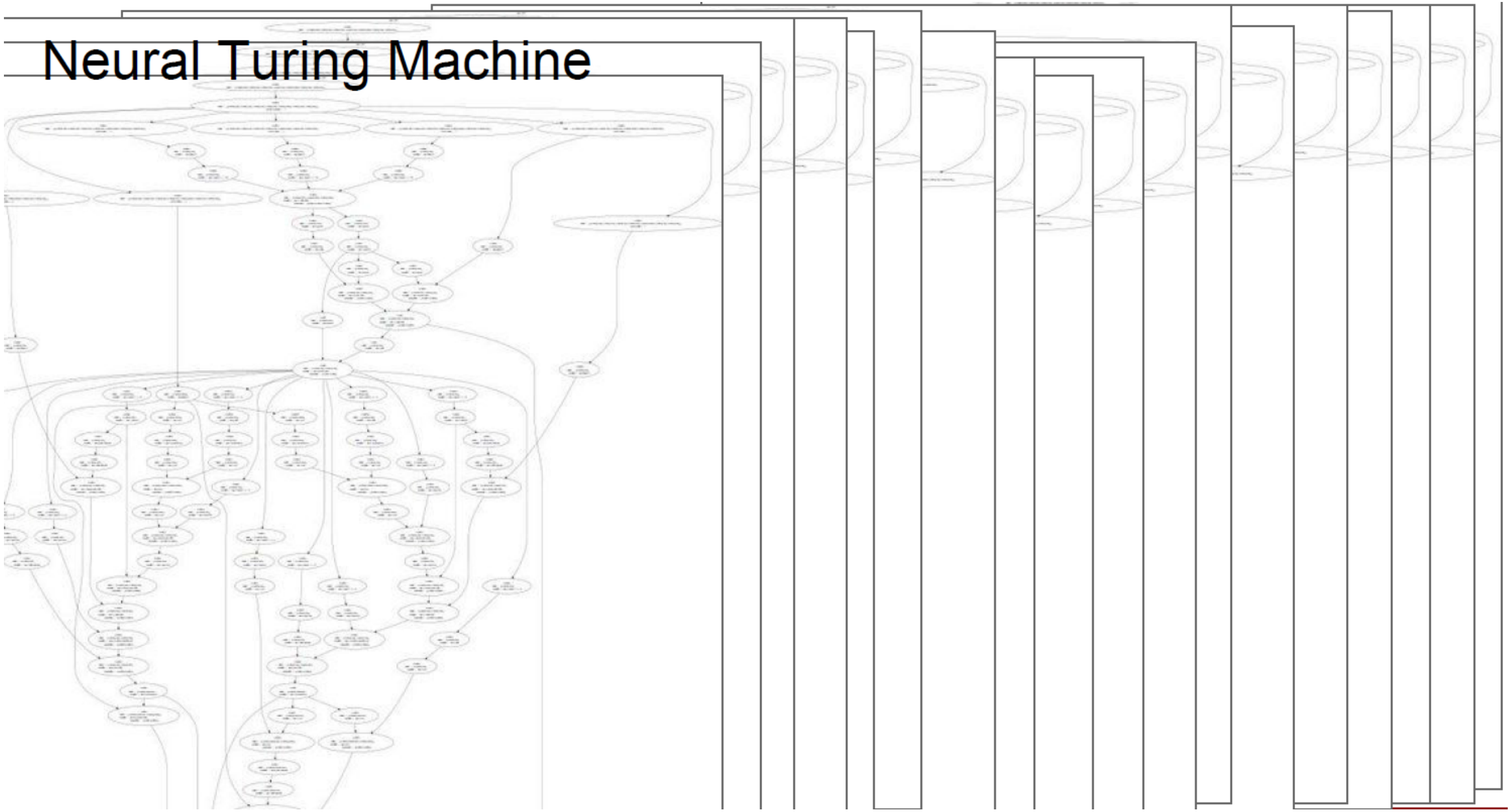
Neural Turing Machine

input tape

loss

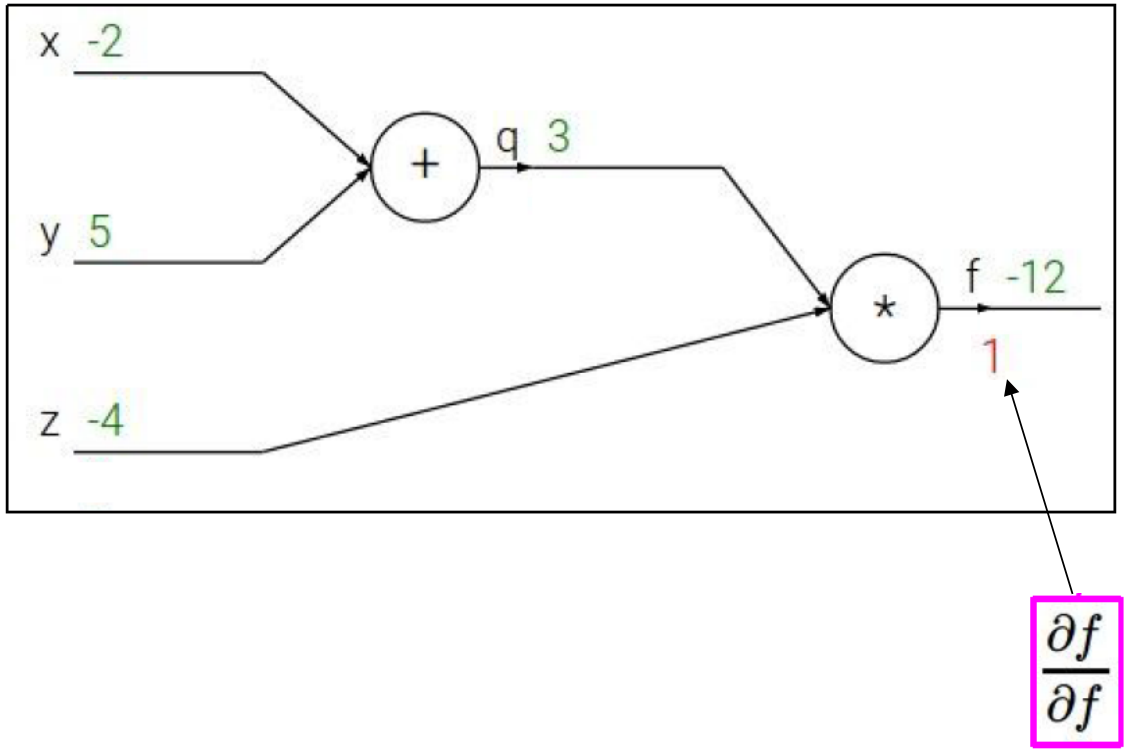


Neural Turing Machine



$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



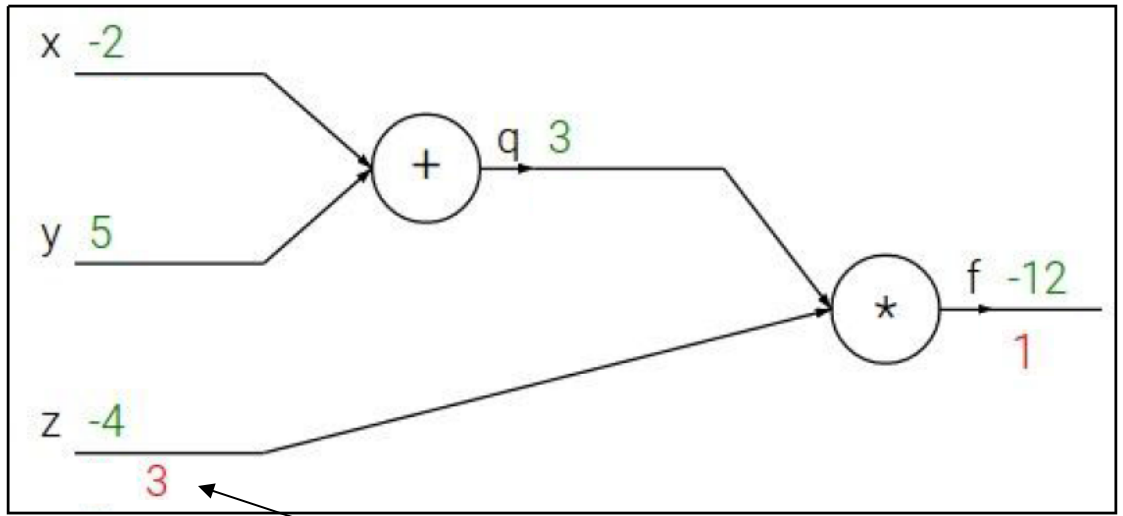
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

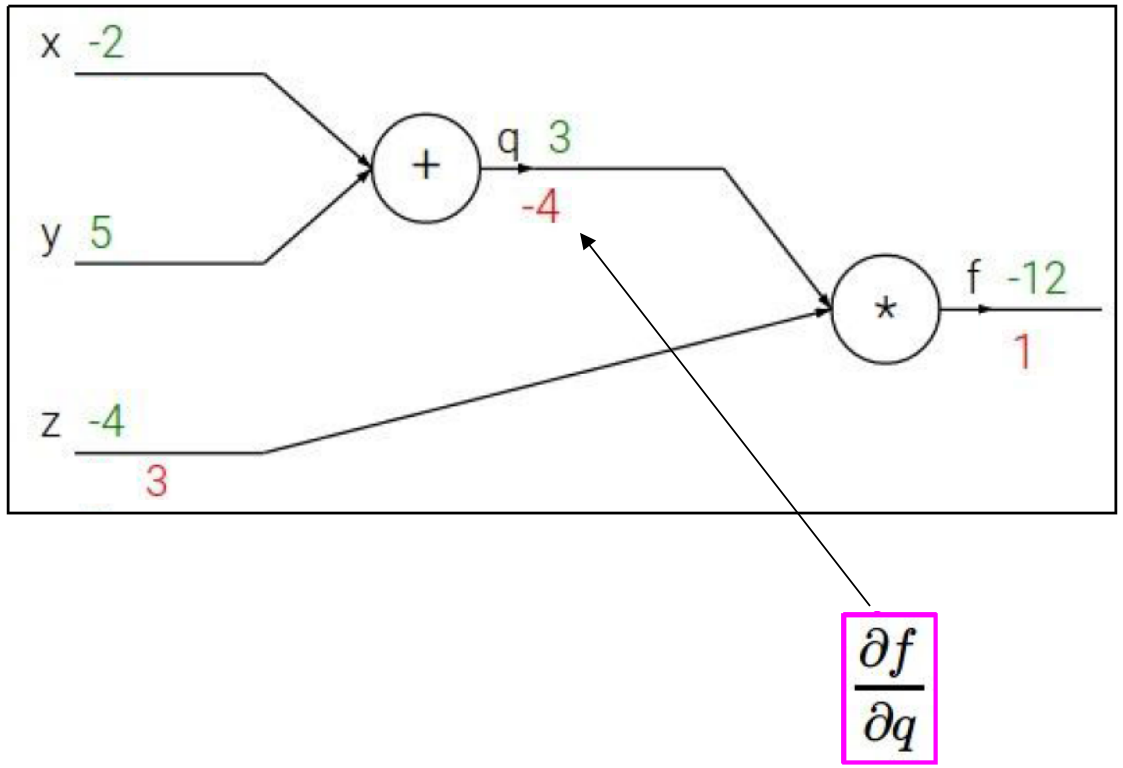
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

$$\frac{\partial f}{\partial z}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



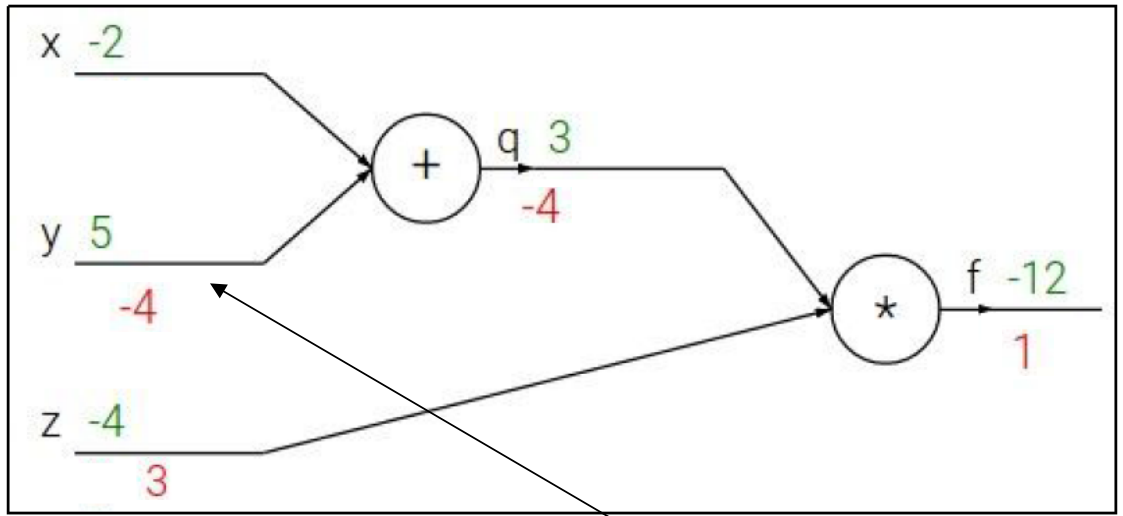
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

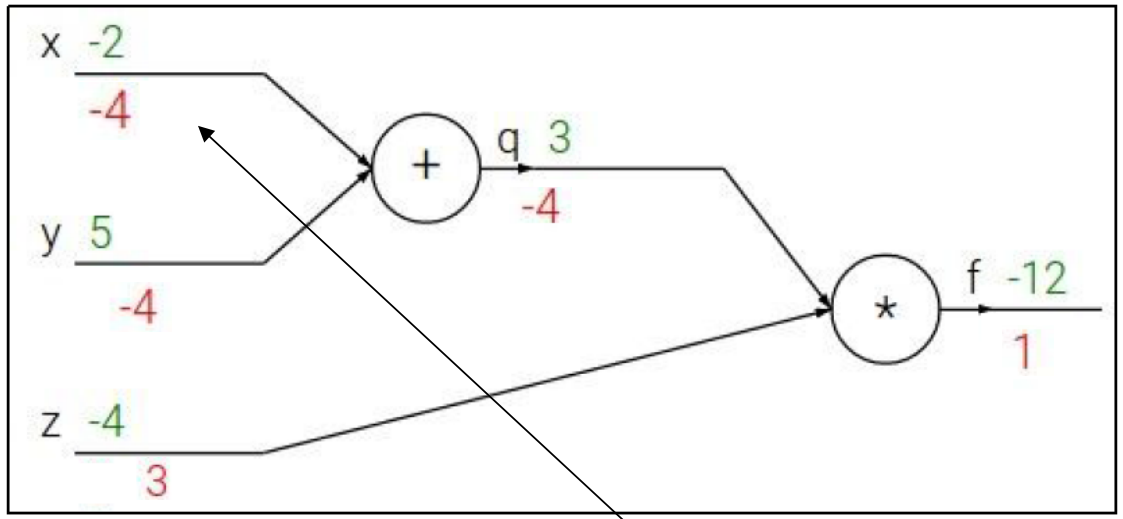
$\frac{\partial f}{\partial y}$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

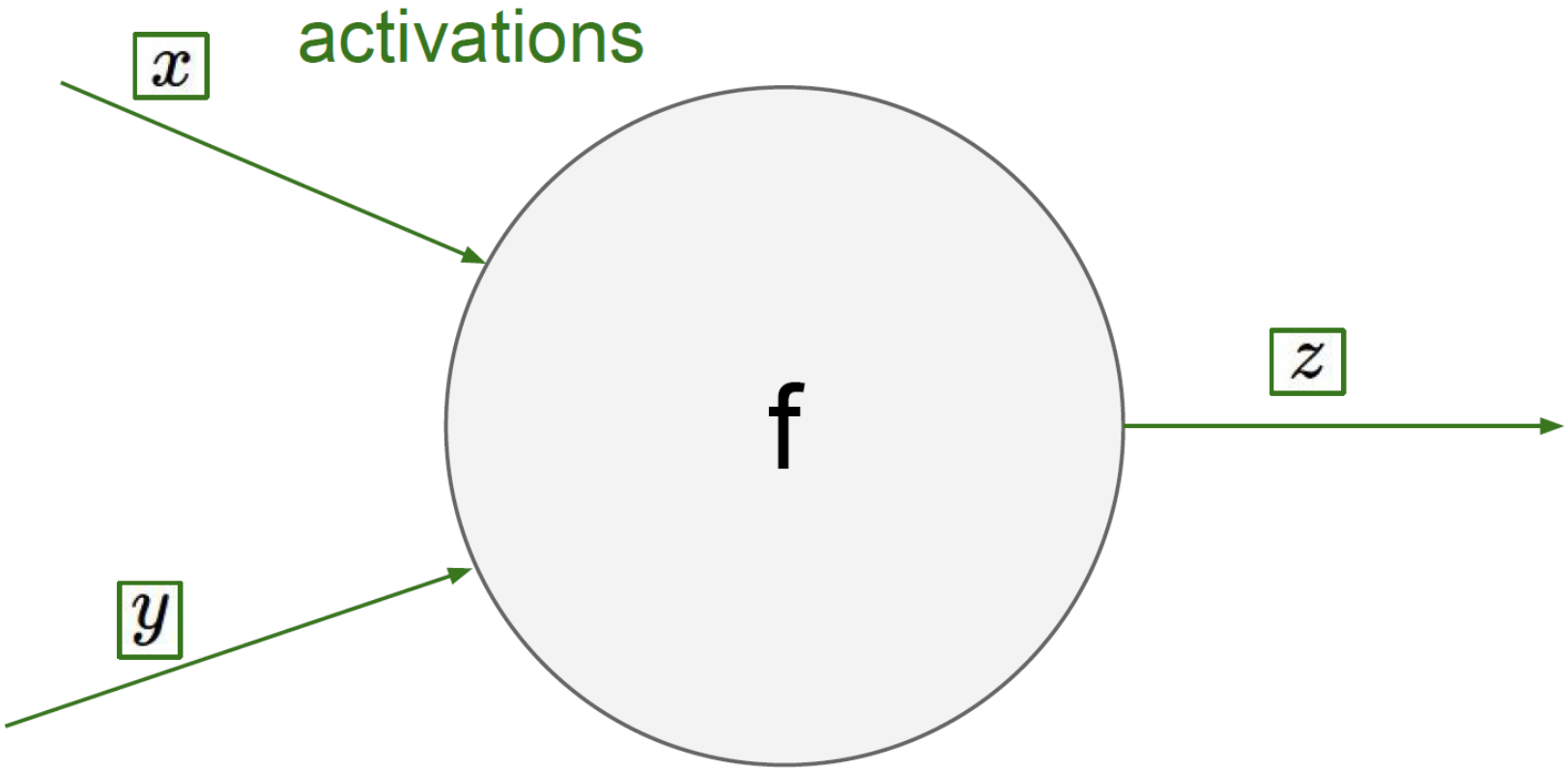
$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

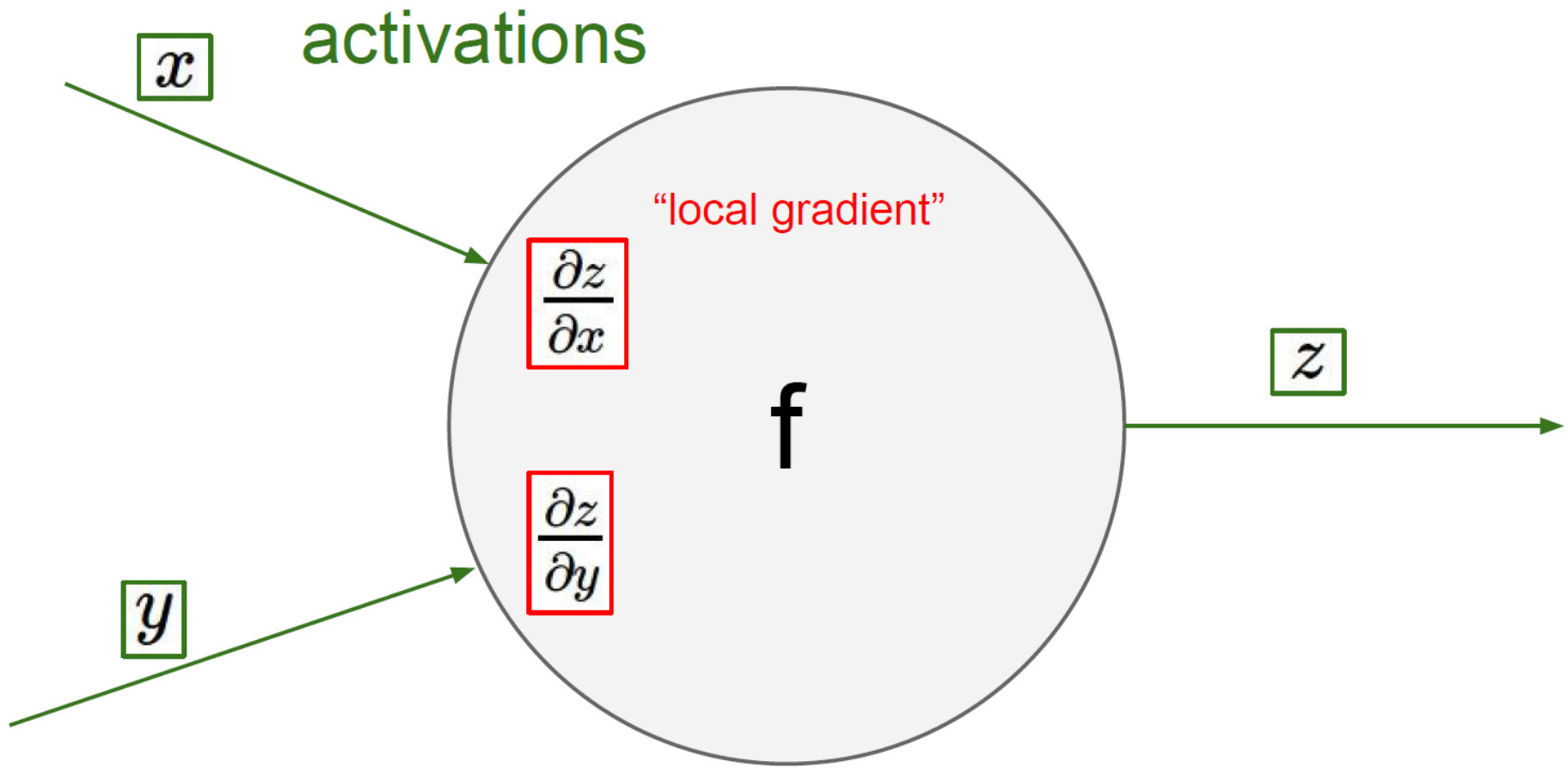
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

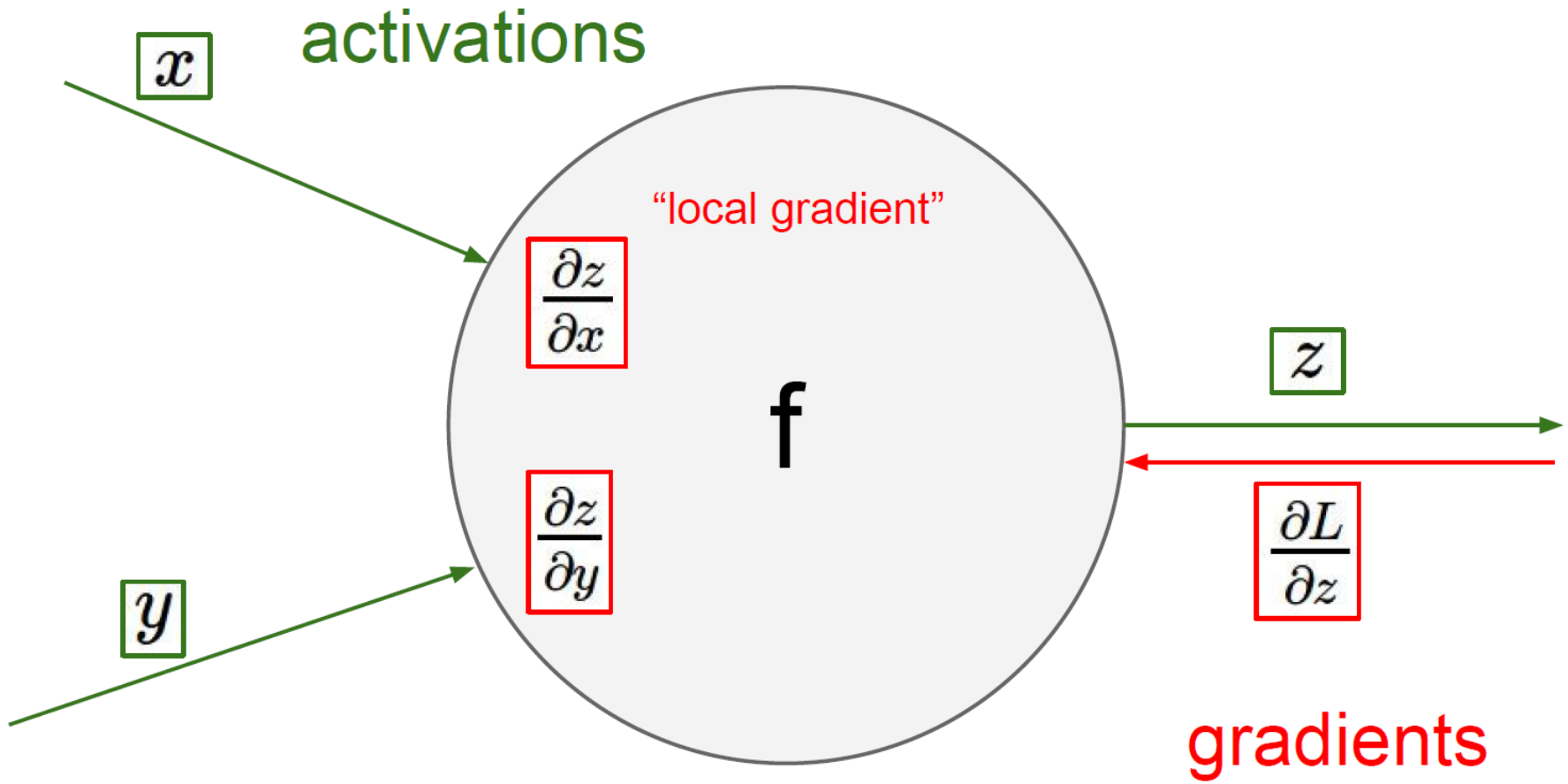
$\frac{\partial f}{\partial x}$

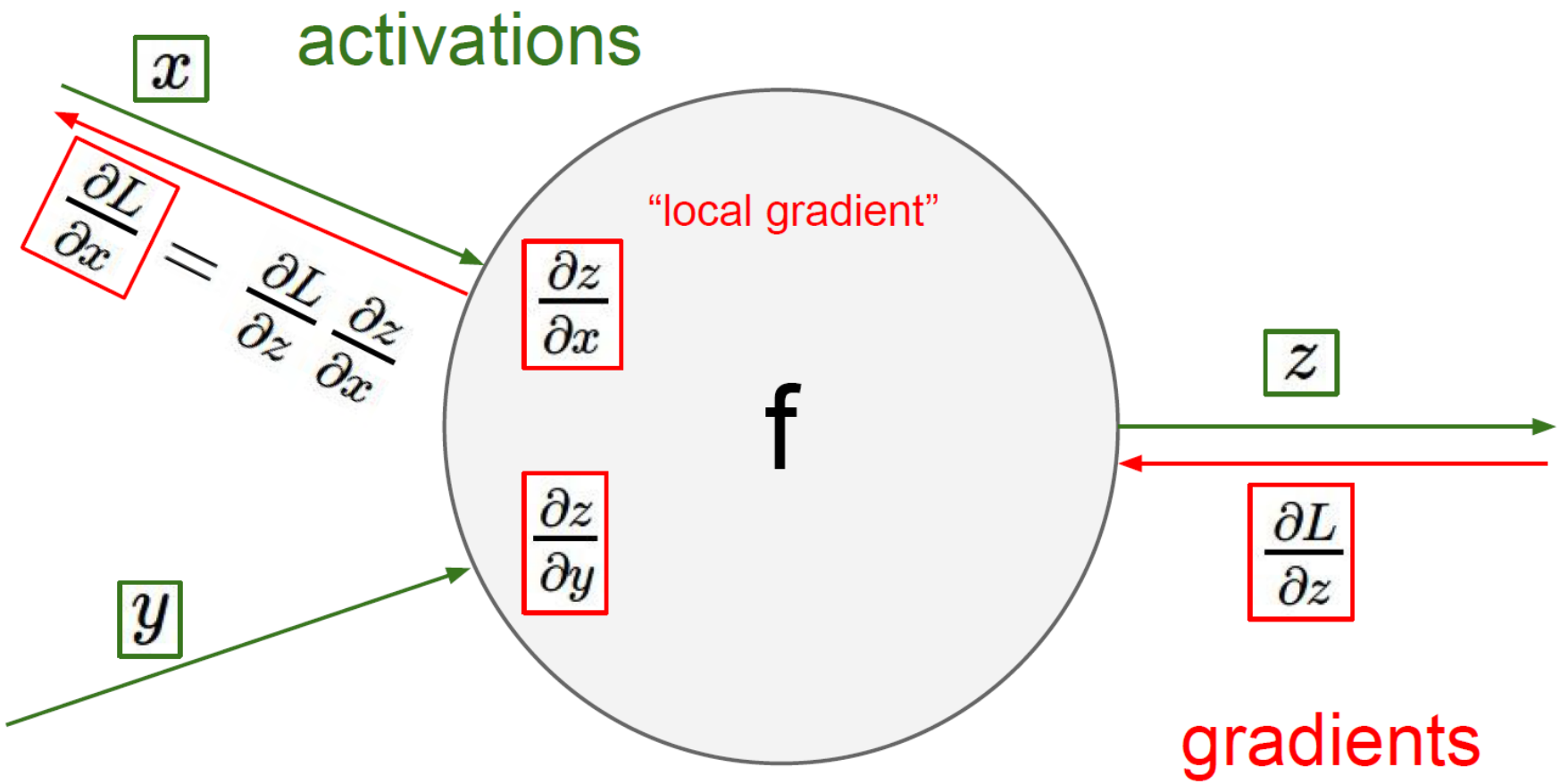
Chain rule:

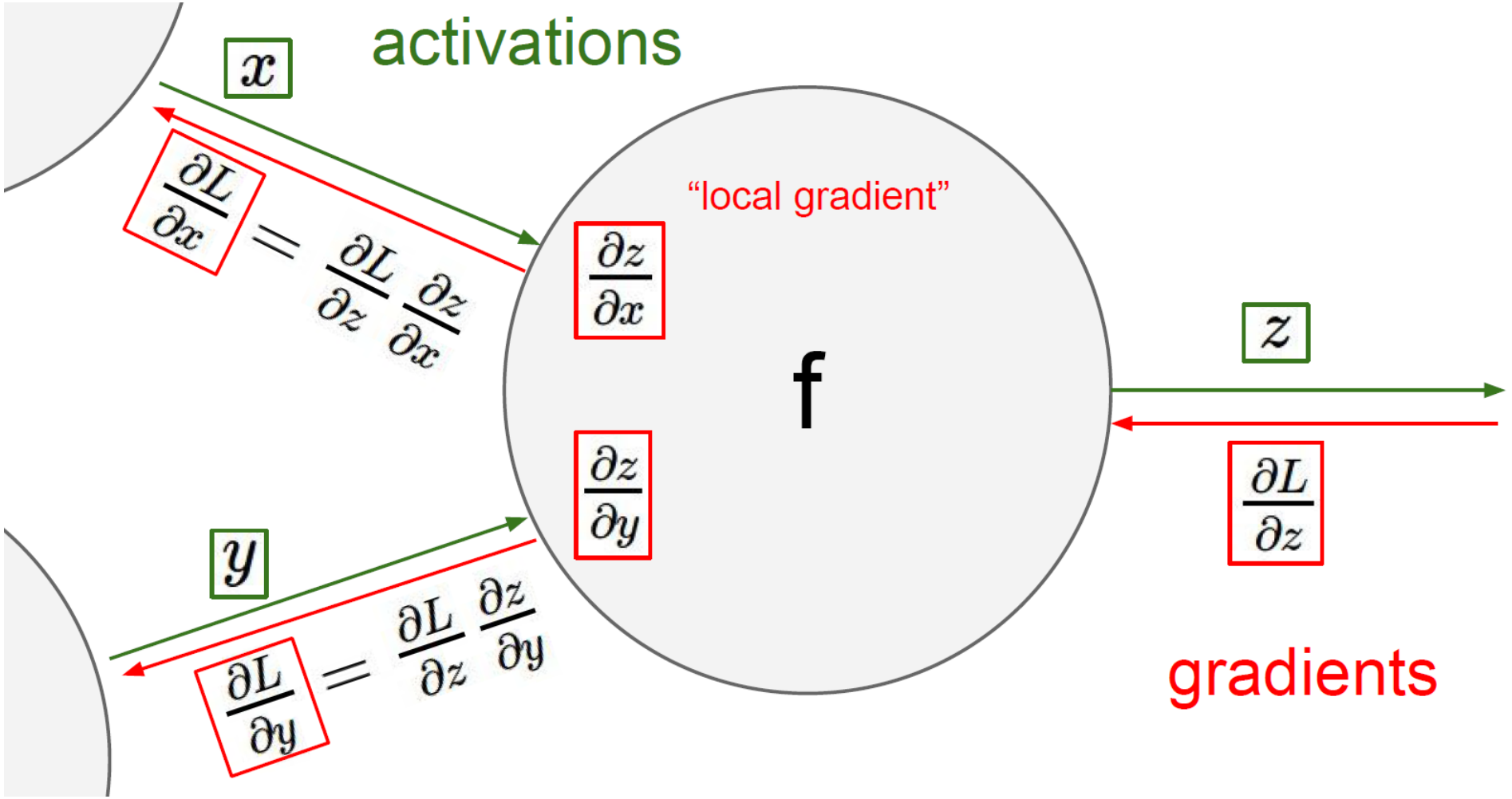
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$


activations
x
y
f
z


activations
x
y
z
f
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$


activations
x
y
"local gradient"
$\frac{\partial z}{\partial x}$
$\frac{\partial z}{\partial y}$
f
z
$\frac{\partial L}{\partial z}$
gradients

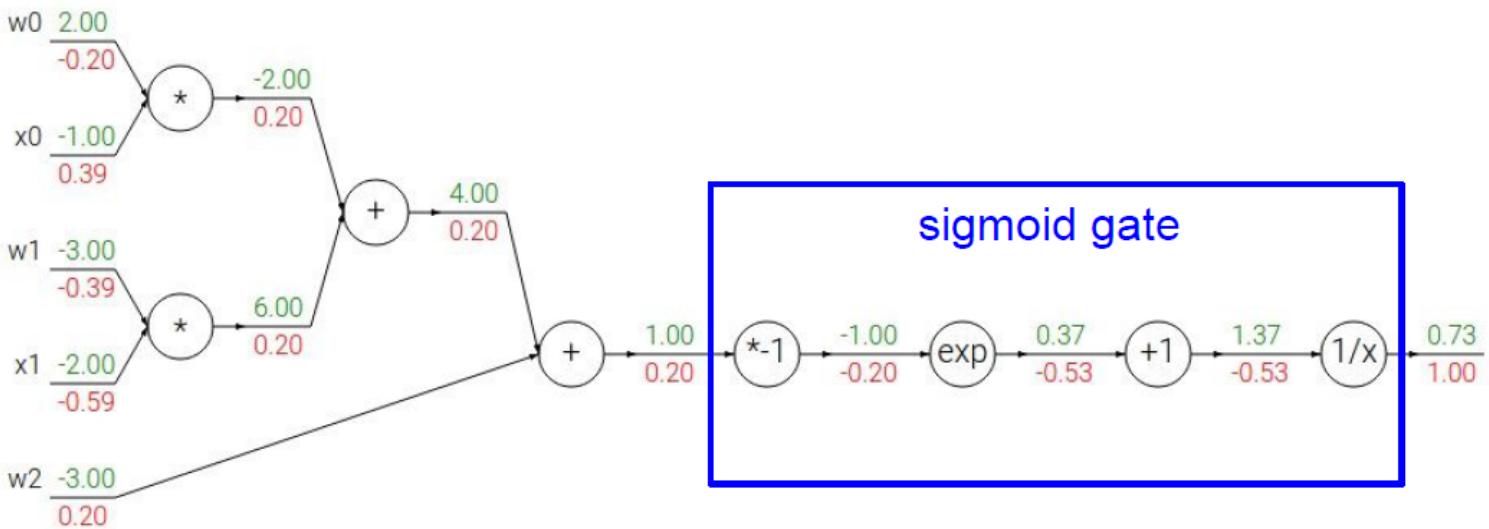

activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
z
$\frac{\partial z}{\partial y}$
$\frac{\partial L}{\partial z}$
y
gradients


activations
x
$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$
"local gradient"
$\frac{\partial z}{\partial x}$
f
$\frac{\partial z}{\partial y}$
z
$\frac{\partial L}{\partial z}$
y
$\frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y}$
gradients

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$ sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$



Keras Structure for Neural Networks

1. The simplest structure of Keras is called '**model**'

```
from keras.models import Sequential

model = Sequential()
```

2. To add a sequence of layers, simply use **'.add()'**

```
from keras.layers import Dense, Activation

model.add(Dense(units=64, input_dim=100))
model.add(Activation('relu'))
model.add(Dense(units=10))
model.add(Activation('softmax'))
```

3. Configure learning process by **.compile()**

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

How to define an optimizer parameter in Keras?

1. You can define the optimizer outside of the .complie() function like this:

```
from keras import optimizers

model = Sequential()
model.add(Dense(64, kernel_initializer='uniform', input_shape=(10,)))
model.add(Activation('tanh'))
model.add(Activation('softmax'))

sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='mean_squared_error', optimizer=sgd)
```

2. Or, you can define it inside the .compile() process:

```
# pass optimizer by name: default parameters will be used
model.compile(loss='mean_squared_error', optimizer='sgd')
```

What are the common optimizers defined in Keras?

- SGD
- ADAM
- RMSprop
- Adagrad
- Adadelta
- Adamax
- Nadam
- TFOptimizer

https://keras.io/optimizers/

What are common loss functions defined in Keras?

- Mean Squared Error
- mean_absolute_error
- mean_absolute_percentage_error
- mean_squared_logarithmic_error
- squared_hinge
- hinge
- categorical_hinge
- binary_crossentropy
- …

https://keras.io/losses/

What is ‘metric’ of configuring a model in Keras? What are the common functions?

- **binary_accuracy**
- **categorical_accuracy**
- **top_k_categorical_accuracy**
- …

What is the main difference of the metric and loss defined in the .compile() process?

https://keras.io/metrics/

Develop Neural Network with Python Keras step by step

1. Load data

```
#load dataset
from sklearn.datasets import load_breast_cancer
bc,tar = load_breast_cancer(return_X_y=True)
Y=tar.reshape(569,)
X = bc.astype(float)
print (X.shape)
print (Y.shape)
```

Develop Neural Network with Python Keras step by step

2. Create Model

```
# create model
model = Sequential()
model.add(Dense(30, input_dim=30, kernel_initializer='normal', activation='relu'))
model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
```

Develop Neural Network with Python Keras step by step

3. compile a model

```
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Develop Neural Network with Python Keras step by step

4. Fit a model

```
history = model.fit(X, y, batch_size=10, epochs=100)
```

Develop Neural Network with Python Keras step by step

5. Evaluate a model

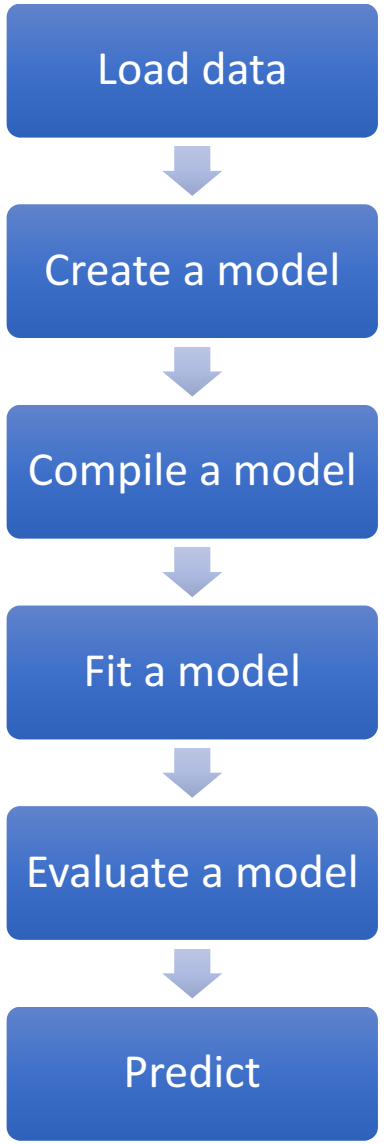
```
loss, accuracy = model.evaluate(X, y)
```

Develop Neural Network with Python Keras step by step

6. Predict with new data

```
predictions = model.predict(x)
```

Summary of Creating a Neural Network Model with Keras



Lets Look at a simple code that summarizes all the steps!

Parameter Tuning using Grid Search

1. batch size and training epochs
2. optimization algorithms
3. learning rate and momentum
4. weight initialization
5. activation functions
6. dropout regularization
7. number of neurons in the hidden layer

Parameter Tuning using Grid Search

```
1  def create_model():
2  ...
3  return model
4
5  model = KerasClassifier(build_fn=create_model)
```

```
1  param_grid = dict(nb_epochs=[10,20,30])
2  grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
3  grid_result = grid.fit(X, Y)
```

Homework 1

- Dataset: Boston house price dataset (available under Sklearn, for more info see: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_boston.html)
- Use the home work description to answer all questions
- Tools: use Jupyter python to write the code, include a brief description for every section of the homework
- Format of submission: PDF (save the code and related description as pdf format and submit on blackboard)
- Deadline: Sept 20th midnight.

Boston house dataset description

Features:

- **CRIM - per capita crime rate by town**
- **ZN - proportion of residential land zoned for lots over 25,000 sq.ft.**
- **INDUS - proportion of non-retail business acres per town.**
- **CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)**
- **NOX - nitric oxides concentration (parts per 10 million)**
- **RM - average number of rooms per dwelling**
- **AGE - proportion of owner-occupied units built prior to 1940**
- **DIS - weighted distances to five Boston employment centres**
- **RAD - index of accessibility to radial highways**
- **TAX - full-value property-tax rate per $10,000**
- **PTRATIO - pupil-teacher ratio by town**
- **B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town**
- **LSTAT - % lower status of the population**
- **MEDV - Median value of owner-occupied homes in $1000's**

http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Homework 1

Part 1: Download Boston home dataset and load it into your program
Hint: The Boston dataset output is price…therefore this is a regression problem! And the output should be a kerasRegressor for model.compile()

Part 2: Define a simple keras model similar to the 'step_by_step_networl.ipynb' code in class, try to find the best hyper parameters using the grid search technique in skitlearn and Keras for the following parameters

a) optimization algorithms
b) learning rate and momentum
c) activation functions
d) dropout regularization

Hint: if the code is slow to run, decrease the dimention of your dataset manually by selecting less number of samples and continue from there

Part 3:

a) Make your dataset standard using the 'standardScalar' function and use all the best parameters you found from previous step to derive new results. Your results should be slightly better.
b) Increase the depth of the network by adding two more hidden layers and record your results, do you see improvement?
c) Increase the width of the network by increasing the number of neurons in the hidden layer and record the results, Is the results better than increasing the depth?

Final Project

- Seletct a dataset that you are comfortable working with (some datasets could be found on Kaggle website https://www.kaggle.com/datasets

- Submit an abstract (one paragraph) along with your names by Sept 20th to nhante2@uic.edu

- A description of the project requirements will be provided online.