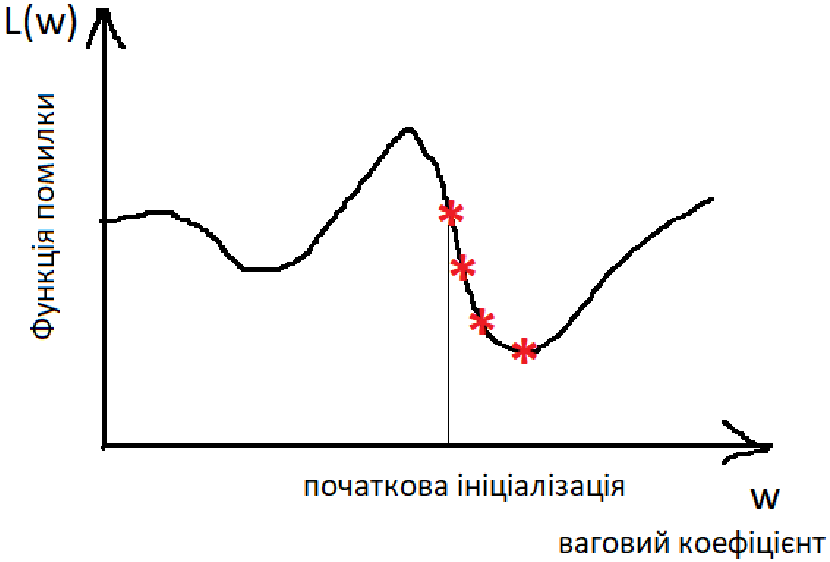
Backpropagation

Loss Function

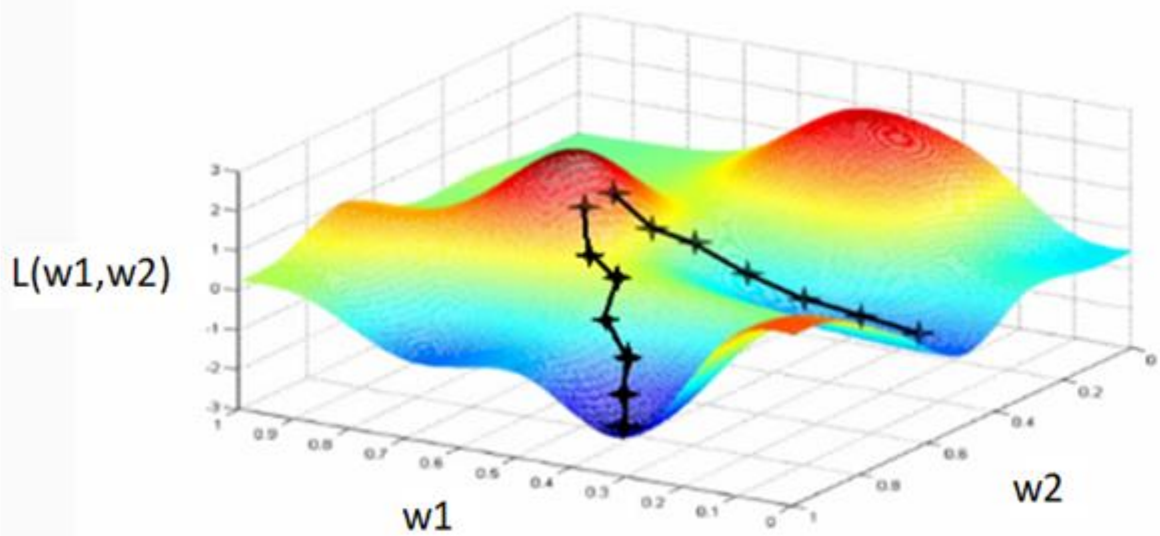
$$L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$$

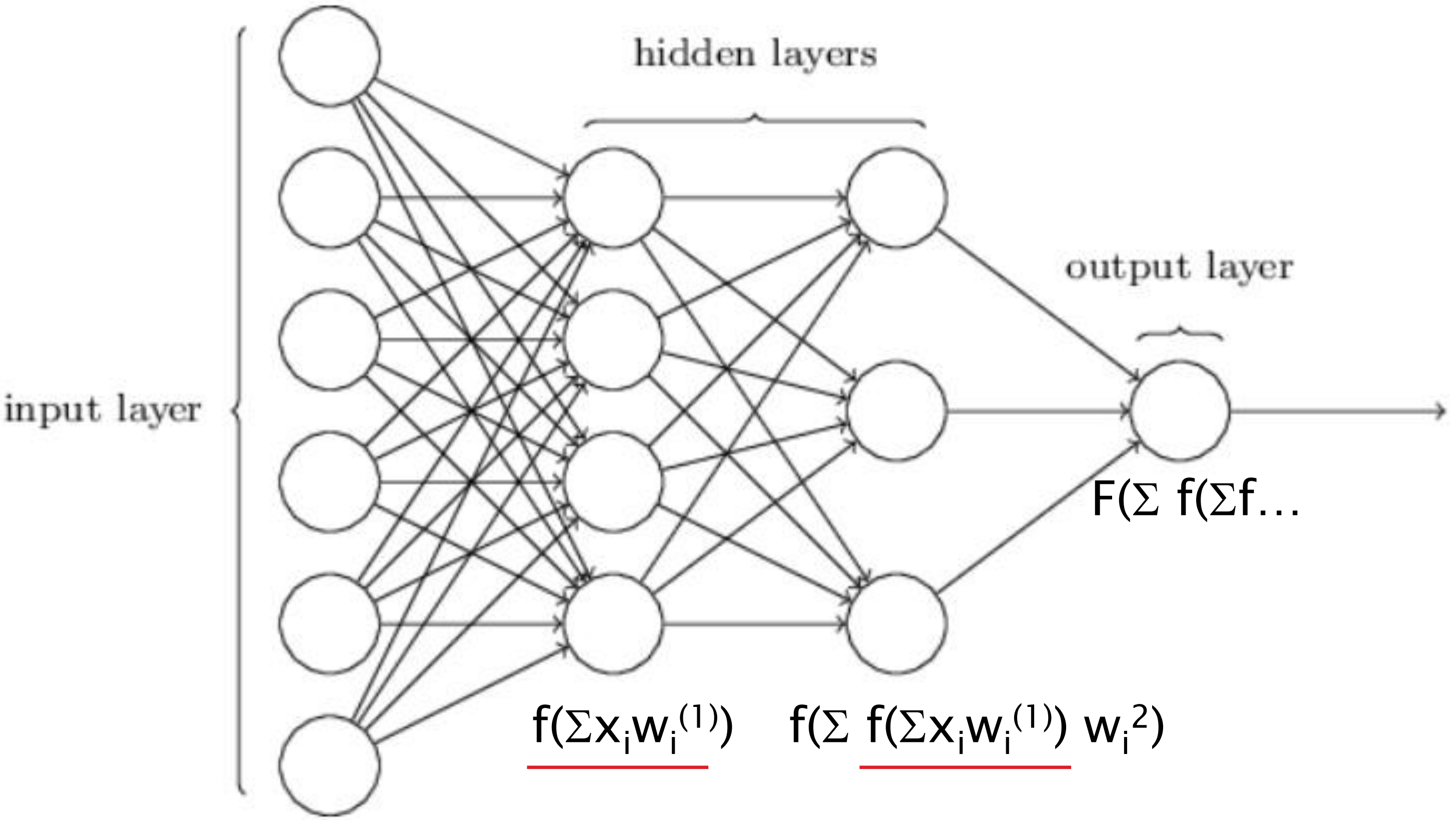
Градієнтний спуск (Gradient descent)



$$\vec{w} = \vec{w} - \eta \overrightarrow{\nabla_w} L$$

$$\vec{b} = \vec{b} - \eta \vec{\nabla}_b L$$





$F(\Sigma\ f(\Sigma f\ldots$

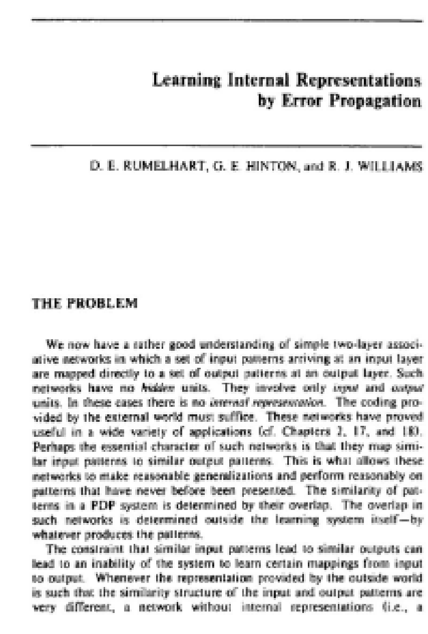
$\underline{f(\Sigma x_i w_i^{(1)})}$ $\quad$ $f(\Sigma\ \underline{f(\Sigma x_i w_i^{(1)})}\ w_i^2)$

1986р: Д.Румельхарт, Дж.Хінтон, Р.Вільямс розробляють обчислювально ефективний алгоритм навчання нейромереж – [метод зворотного поширення помилки].

Джефрі Хінтон

Девід Румельхарт

Рональд Вільямс



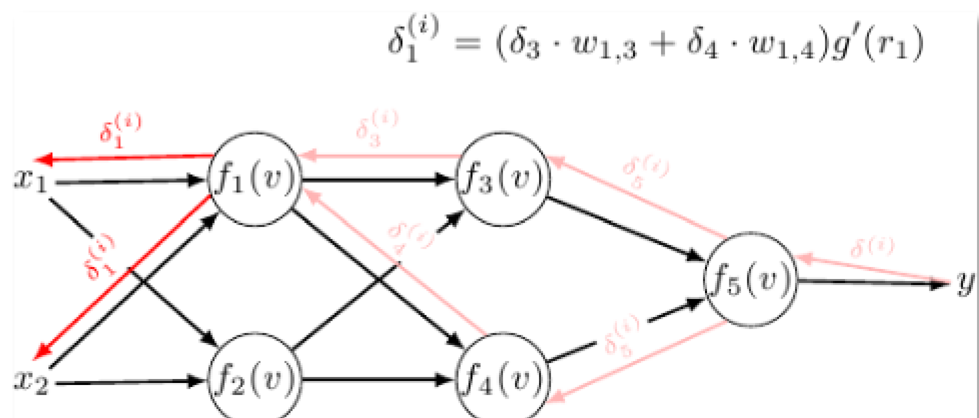
Learning Internal Representations by Error Propagation

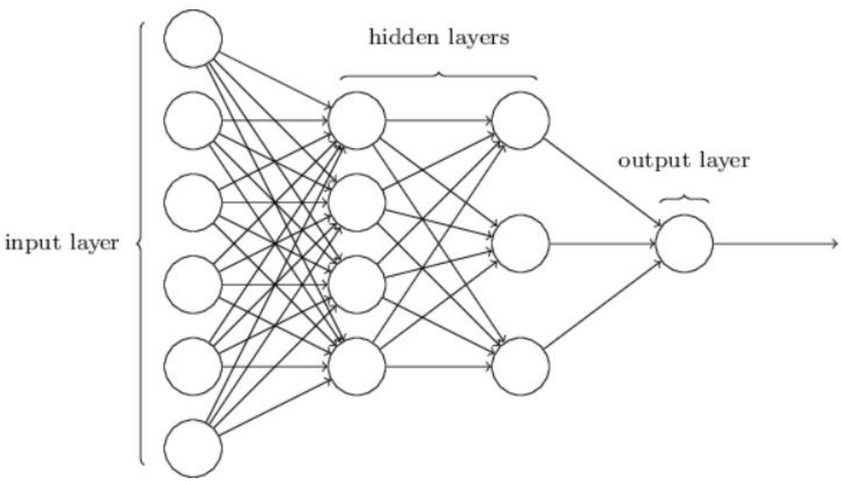
D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS

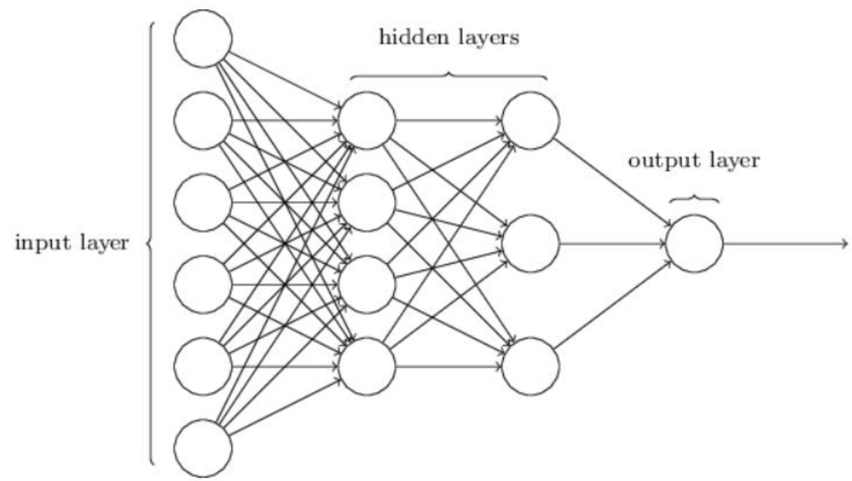
THE PROBLEM

We now have a rather good understanding of simple two-layer associative networks in which a set of input patterns arriving at an input layer are mapped directly to a set of output patterns at an output layer. Such networks have no *hidden* units. They involve only *input* and *output* units. In these cases there is no *internal representation*. The coding provided by the external world must suffice. These networks have proved useful in a wide variety of applications (cf. Chapters 2, 17, and 18). Perhaps the essential character of such networks is that they map similar input patterns to similar output patterns. This is what allows these networks to make reasonable generalizations and perform reasonably on patterns that have never before been presented. The similarity of patterns in a PDP system is determined by their overlap. The overlap in such networks is determined outside the learning system itself—by whatever produces the patterns.

The constraint that similar input patterns lead to similar outputs can lead to an inability of the system to learn certain mappings from input to output. Whenever the representation provided by the outside world is such that the similarity structure of the input and output patterns are very different, a network without internal representations (i.e., a







forward pass

прямий прохід

обчислюємо значення нейронів та функцію втрат

backward pass

зворотній прохід

обчислюємо похідні та корегуємо вагові коефіцієнти

Chain rule

$f(g(x))$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

upstream gradient

local gradient

$f(g(x(a)))$

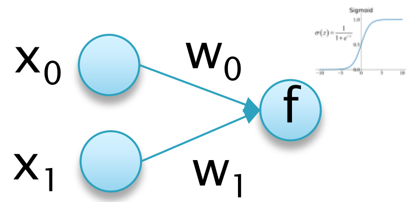
$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x} \cdot \frac{\partial x}{\partial a} = \frac{\partial f}{\partial x} \cdot \frac{\partial x}{\partial a}$$

$f(g(x(a, b)))$

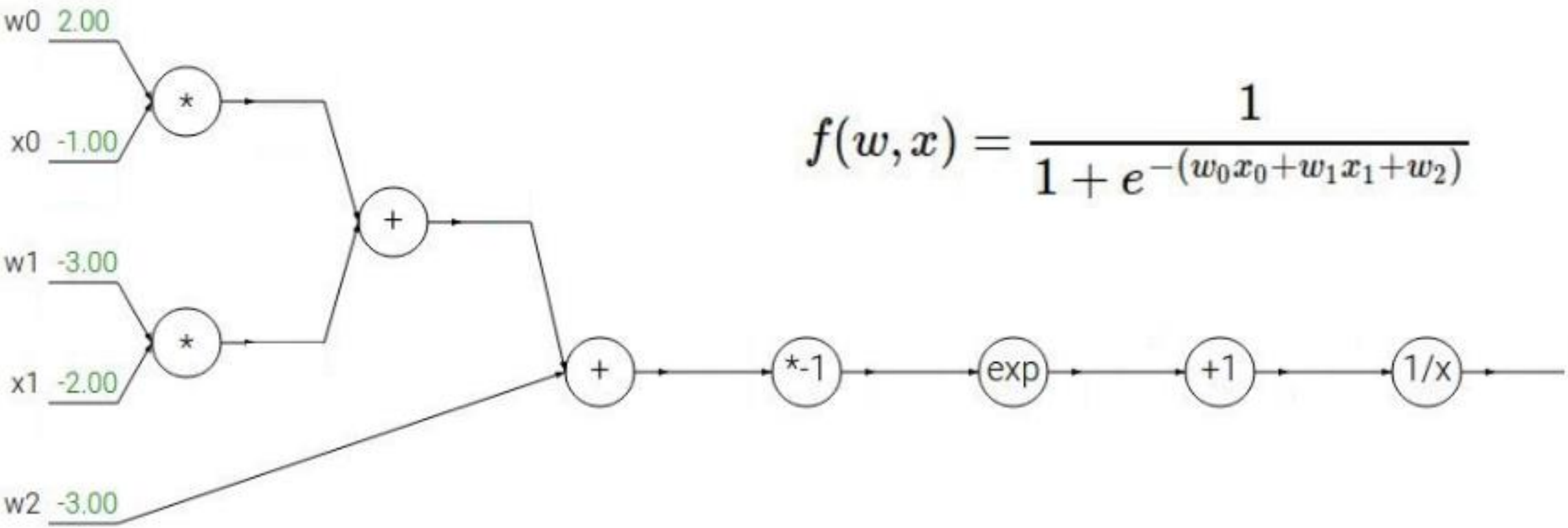
$$\frac{\partial f}{\partial a} = \boxed{\frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}} \cdot \frac{\partial x}{\partial a}$$

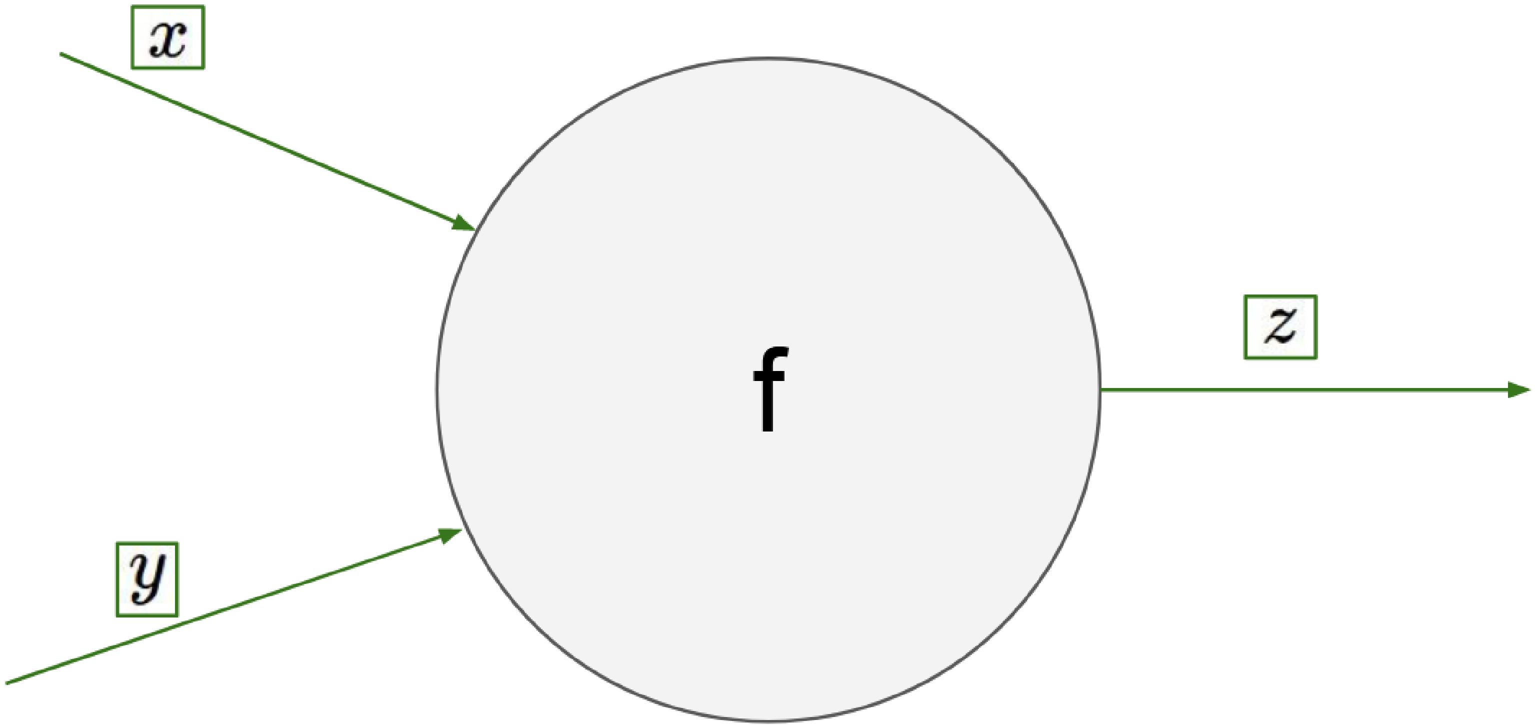
$$\frac{\partial f}{\partial b} = \boxed{\frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}} \cdot \frac{\partial x}{\partial b}$$

Computational graph

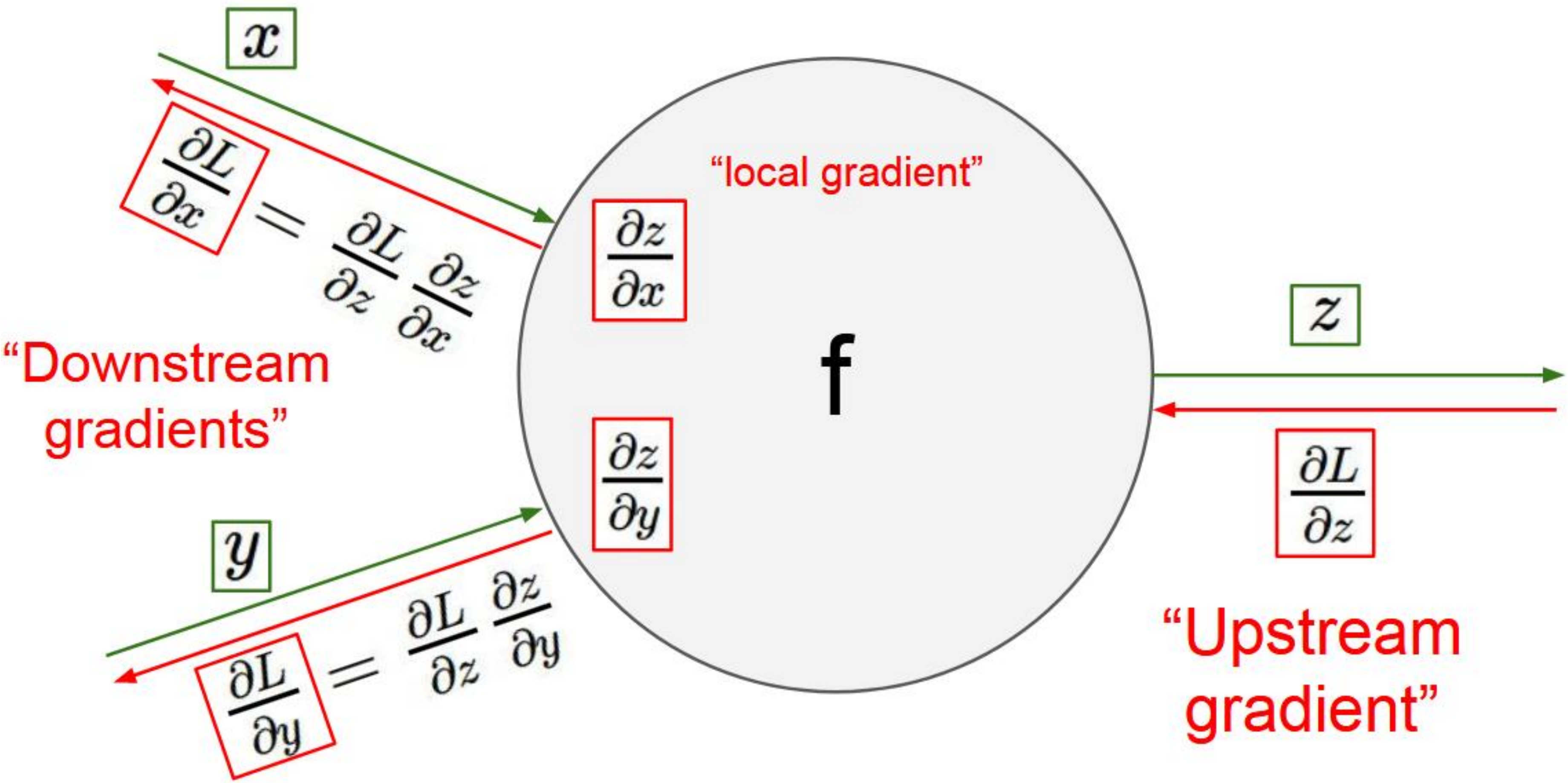


$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

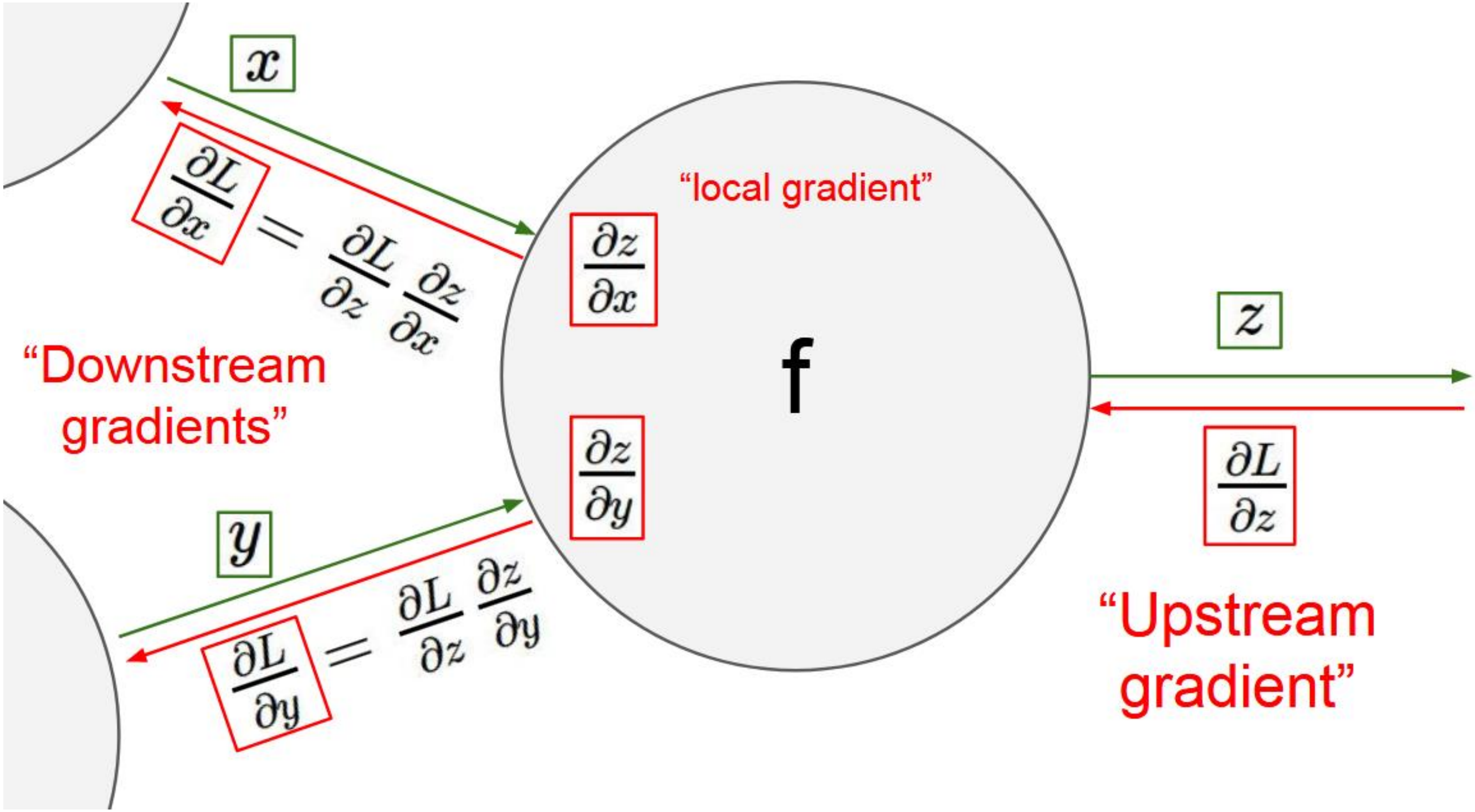




http://cs231n.stanford.edu/slides/2021/lecture_4.pdf



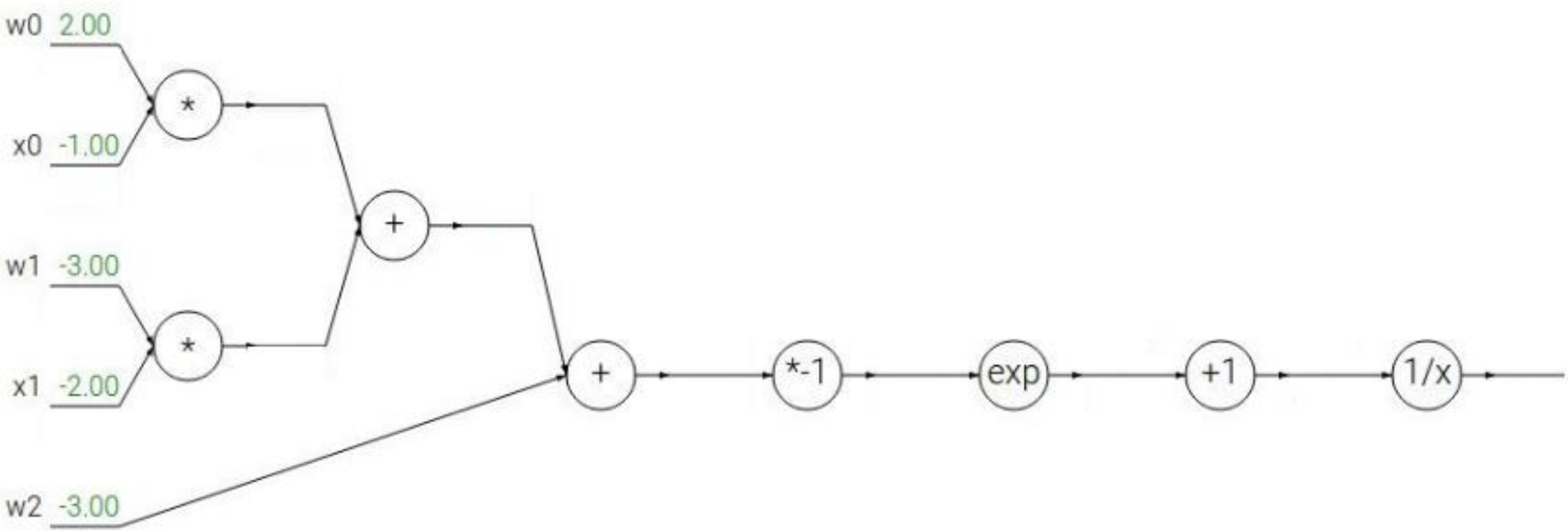
http://cs231n.stanford.edu/slides/2021/lecture_4.pdf



http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

Another example:

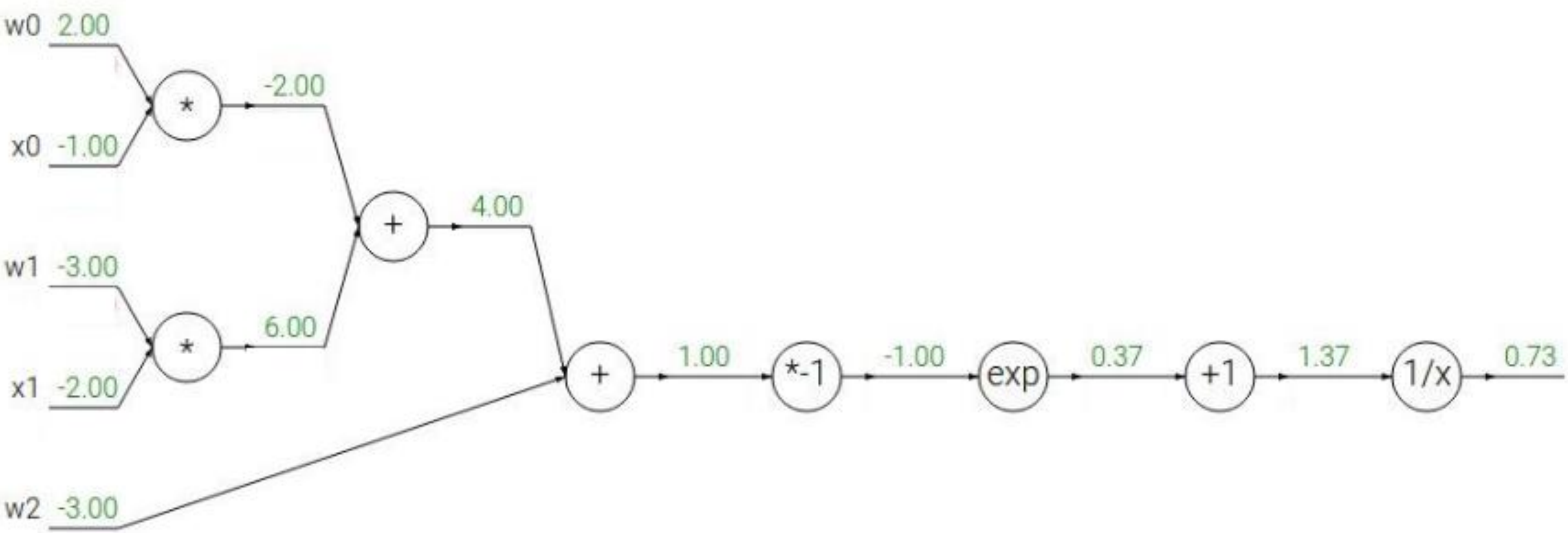
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

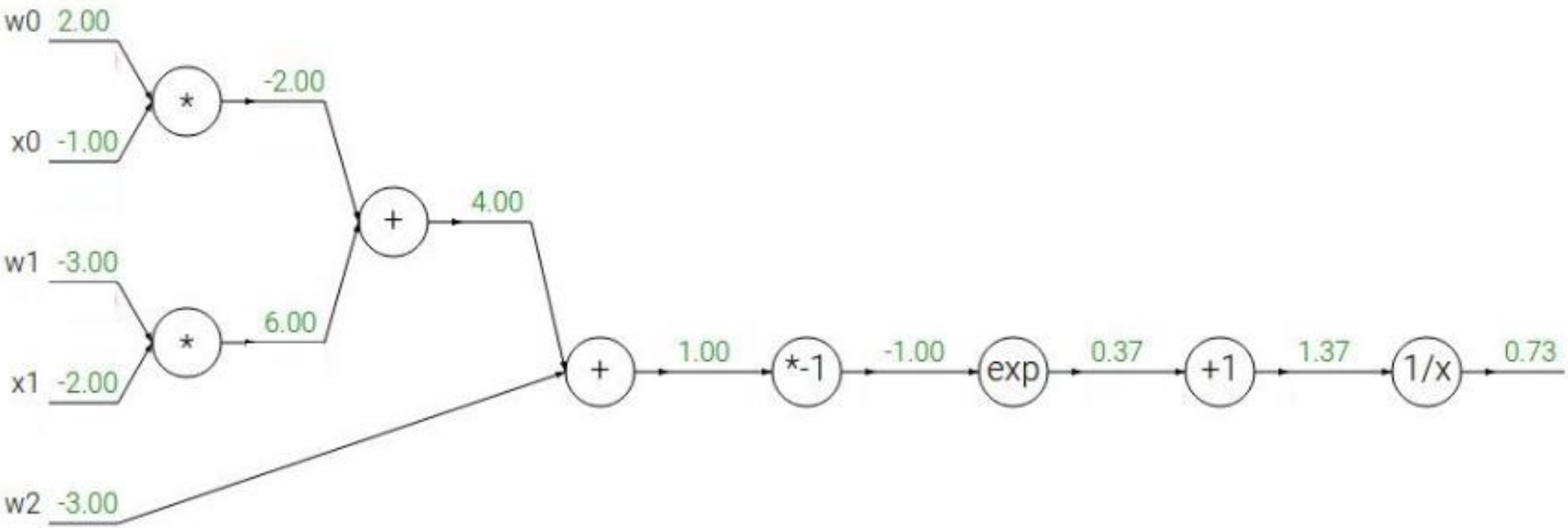
Another example:

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



http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

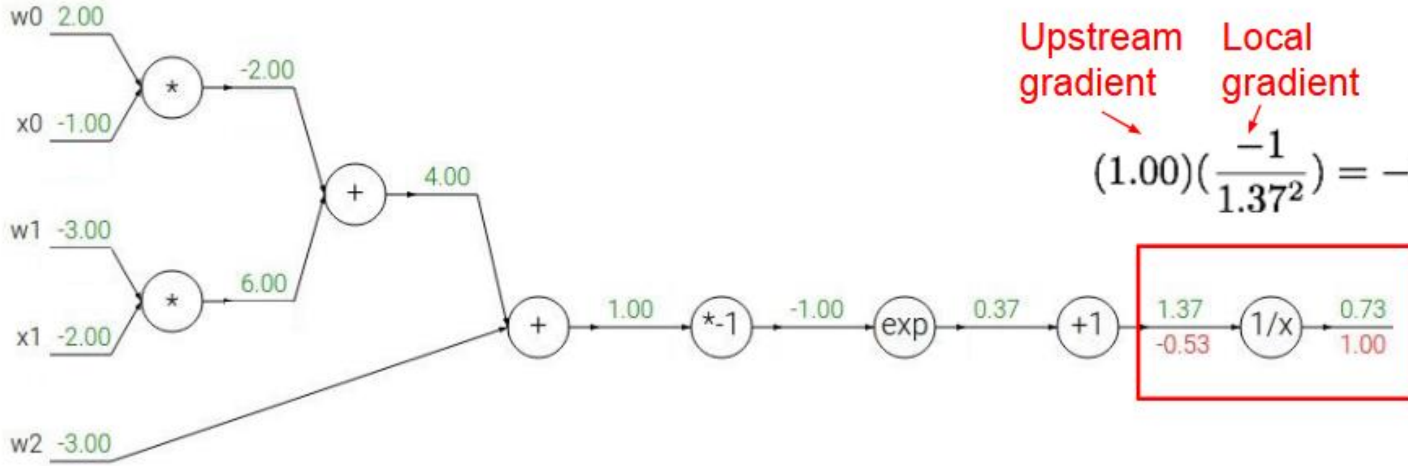
Another example: $f(w,x)=\frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



$f(x)=e^x \quad \rightarrow \quad \frac{df}{dx}=e^x$	$f(x)=\frac{1}{x} \quad \rightarrow \quad \frac{df}{dx}=-1/x^2$
$f_a(x)=ax \quad \rightarrow \quad \frac{df}{dx}=a$	$f_c(x)=c+x \quad \rightarrow \quad \frac{df}{dx}=1$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

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



$(1.00)(\frac{-1}{1.37^2}) = -0.53$

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

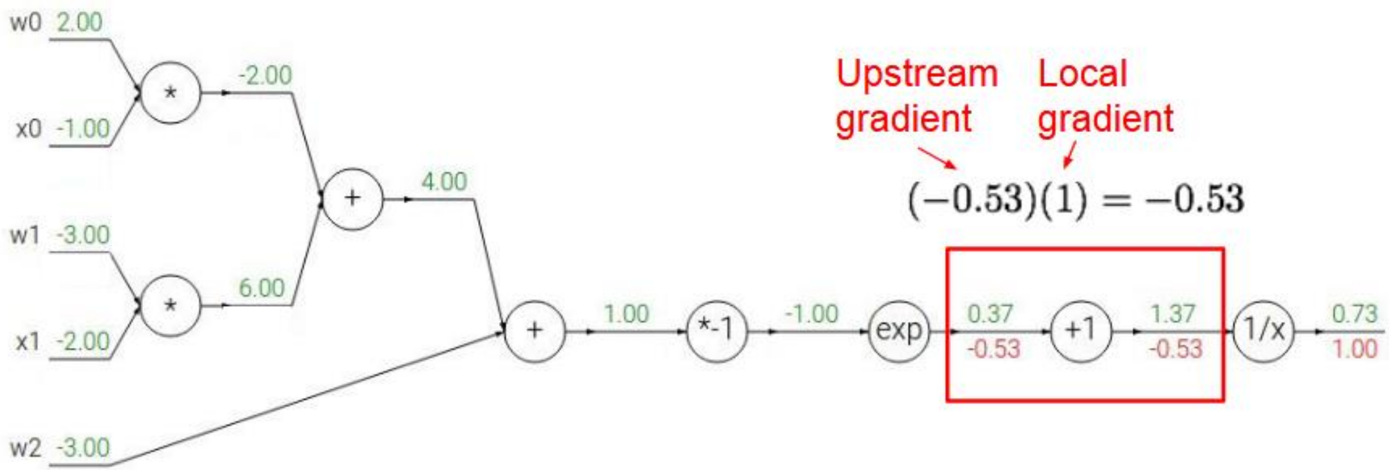
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

$\boxed{f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2}$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

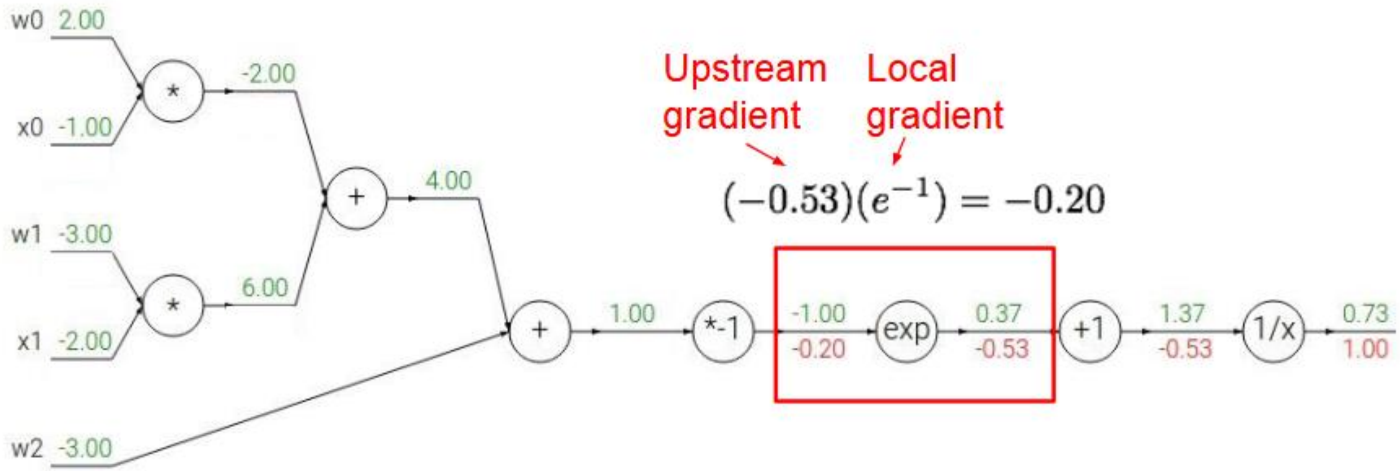
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$\boxed{f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1}$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

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



$$\boxed{f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x}$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

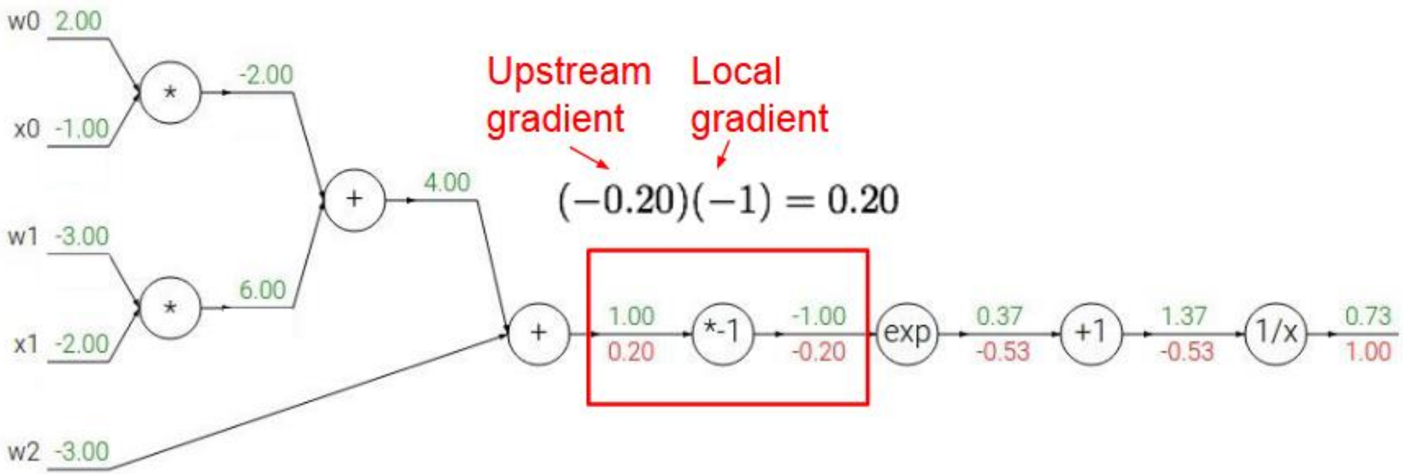
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

Another example:

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



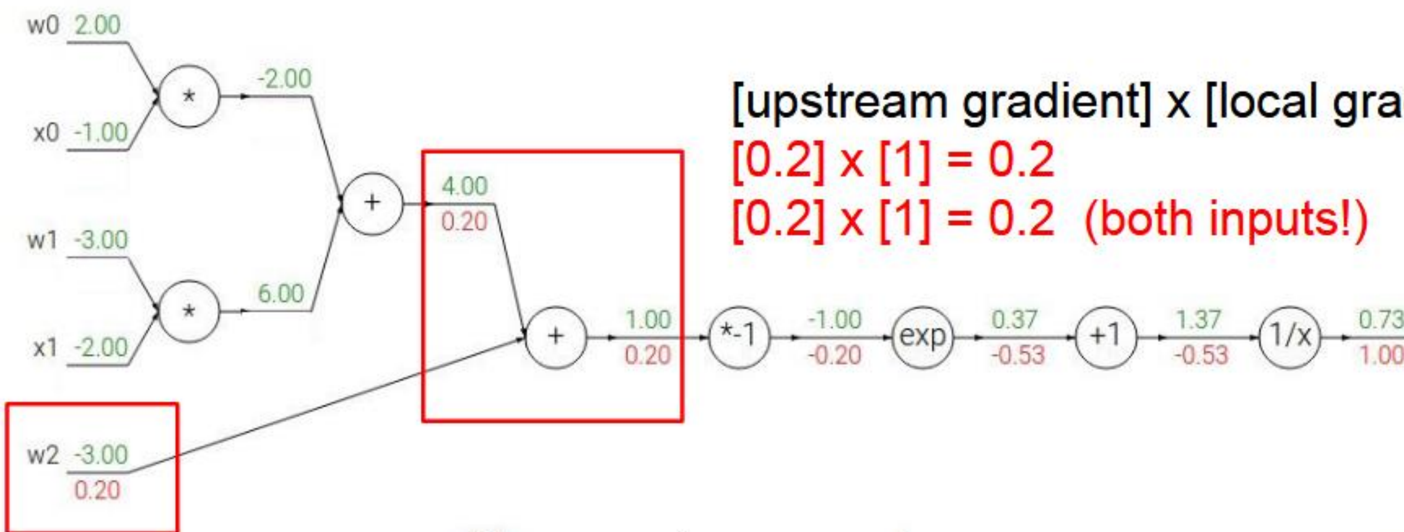
$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$\boxed{f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a}$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

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



[upstream gradient] x [local gradient]

[0.2] x [1] = 0.2

[0.2] x [1] = 0.2 (both inputs!)

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

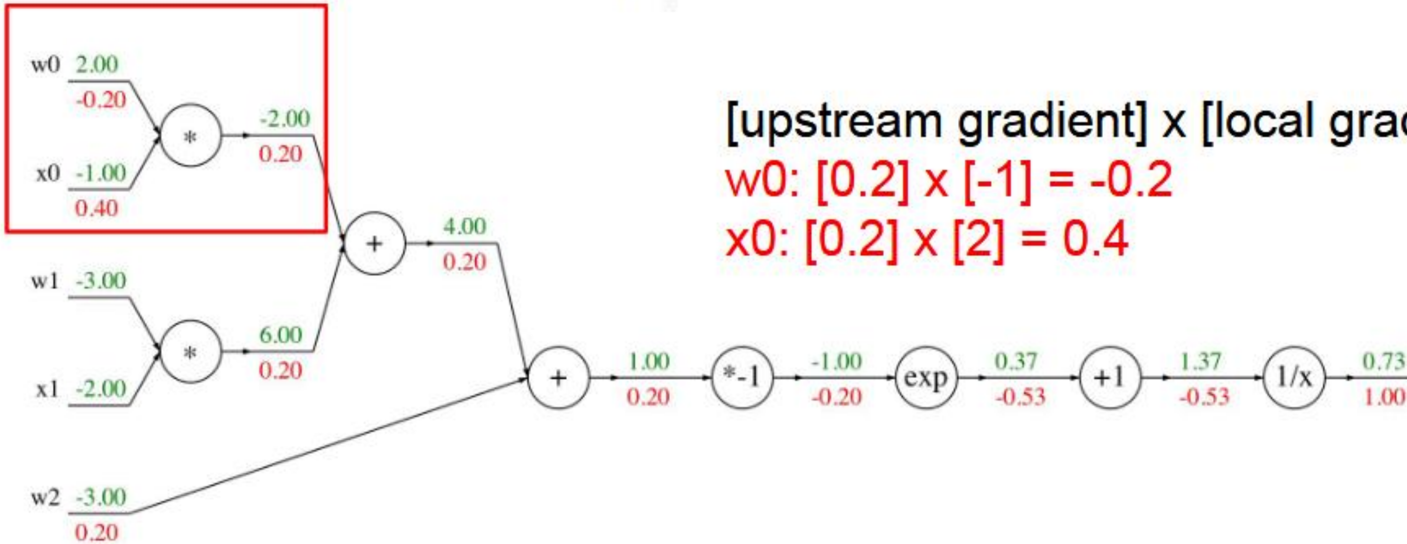
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

Another example:

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



[upstream gradient] x [local gradient]

w0: [0.2] x [-1] = -0.2

x0: [0.2] x [2] = 0.4

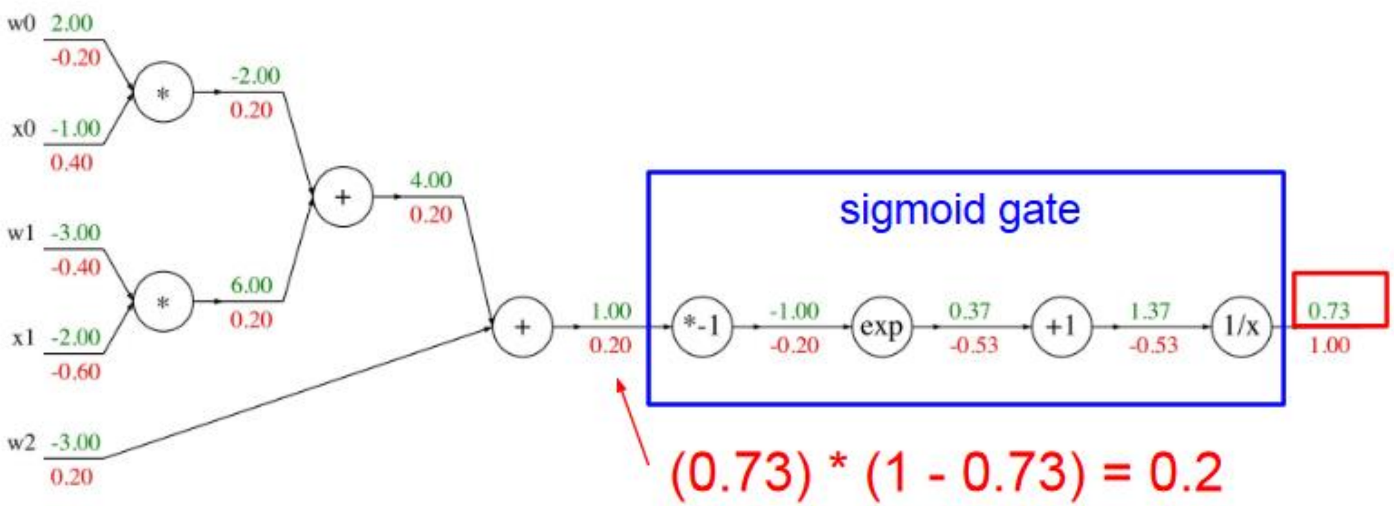
$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

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

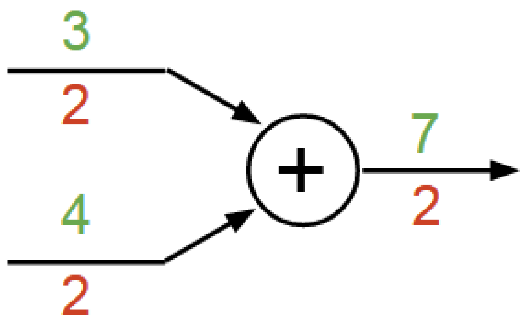
$$\boxed{\sigma(x) = \frac{1}{1+e^{-x}}}$$ sigmoid function

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

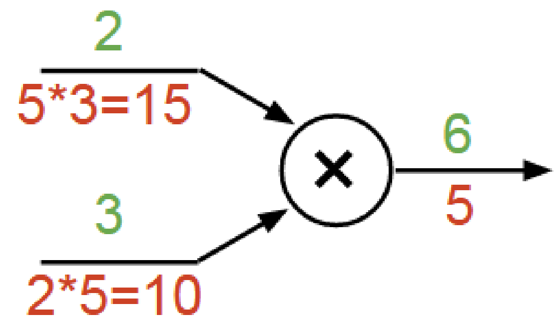


Patterns in gradient flow

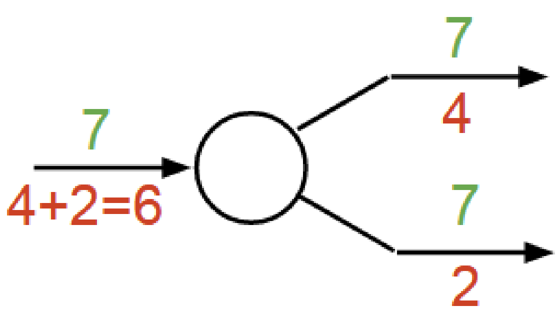
add gate: gradient distributor



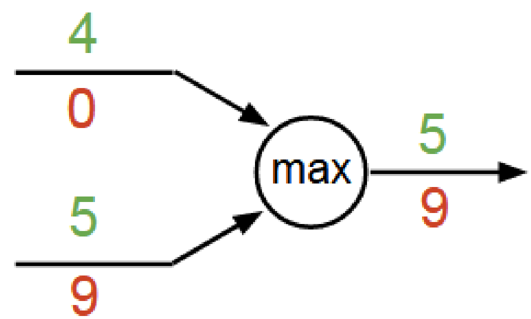
mul gate: “swap multiplier”



copy gate: gradient adder



max gate: gradient router

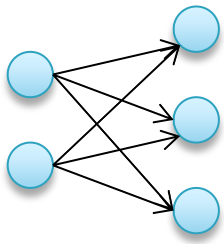


http://cs231n.stanford.edu/slides/2021/lecture_4.pdf

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}$$

$$\frac{\partial L}{\partial W} - ? \qquad \frac{\partial L}{\partial X} - ?$$

$$\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}$$



$$\begin{pmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \end{pmatrix}$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial W} \qquad \frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X}$$

$$\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix} \qquad \frac{\partial L}{\partial X} = \begin{pmatrix} \frac{\partial L}{\partial x_{11}} & \frac{\partial L}{\partial x_{12}} \\ \frac{\partial L}{\partial x_{21}} & \frac{\partial L}{\partial x_{22}} \end{pmatrix} \qquad \frac{\partial L}{\partial Y} = \begin{pmatrix} \frac{\partial L}{\partial y_{11}} & \frac{\partial L}{\partial y_{12}} & \frac{\partial L}{\partial y_{13}} \\ \frac{\partial L}{\partial y_{21}} & \frac{\partial L}{\partial y_{22}} & \frac{\partial L}{\partial y_{23}} \end{pmatrix}$$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial w_{11}} = \begin{pmatrix} \frac{\partial L}{\partial y_{11}} & \frac{\partial L}{\partial y_{12}} & \frac{\partial L}{\partial y_{13}} \\ \frac{\partial L}{\partial y_{21}} & \frac{\partial L}{\partial y_{22}} & \frac{\partial L}{\partial y_{23}} \end{pmatrix} \begin{pmatrix} x_{11} & 0 & 0 \\ x_{21} & 0 & 0 \end{pmatrix} = \frac{\partial L}{\partial y_{11}} x_{11} + \frac{\partial L}{\partial y_{21}} x_{21}$$

$$\frac{\partial Y}{\partial w_{11}} = \begin{pmatrix} x_{11} & 0 & 0 \\ x_{21} & 0 & 0 \end{pmatrix} \quad \frac{\partial Y}{\partial w_{12}} = \begin{pmatrix} 0 & x_{11} & 0 \\ 0 & x_{21} & 0 \end{pmatrix} \quad \frac{\partial Y}{\partial w_{13}} = \begin{pmatrix} 0 & 0 & x_{11} \\ 0 & 0 & x_{21} \end{pmatrix}$$

$$\frac{\partial Y}{\partial w_{21}} = \begin{pmatrix} x_{12} & 0 & 0 \\ x_{22} & 0 & 0 \end{pmatrix} \quad \frac{\partial Y}{\partial w_{22}} = \begin{pmatrix} 0 & x_{12} & 0 \\ 0 & x_{22} & 0 \end{pmatrix} \quad \frac{\partial Y}{\partial w_{23}} = \begin{pmatrix} 0 & 0 & x_{12} \\ 0 & 0 & x_{22} \end{pmatrix}$$

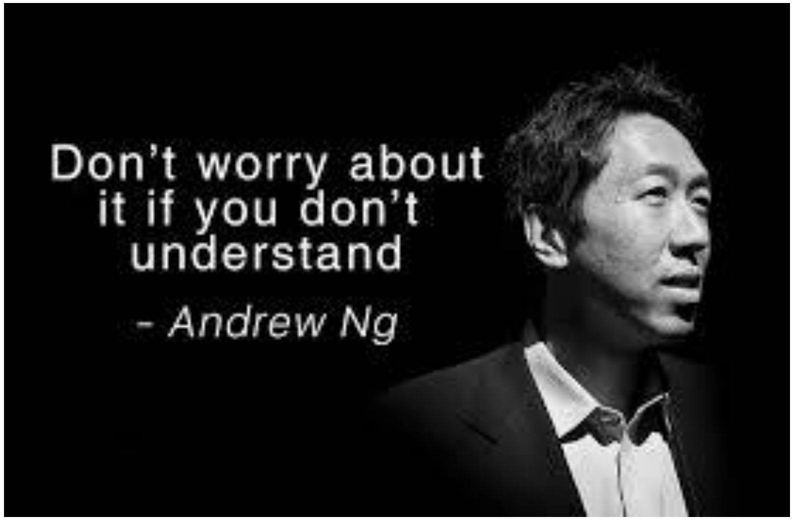
$$\frac{\partial L}{\partial W} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix} = \begin{pmatrix} (\frac{\partial L}{\partial y_{11}} x_{11} + \frac{\partial L}{\partial y_{21}} x_{21}) & (\frac{\partial L}{\partial y_{12}} x_{11} + \frac{\partial L}{\partial y_{22}} x_{21}) & (\frac{\partial L}{\partial y_{13}} x_{11} + \frac{\partial L}{\partial y_{23}} x_{21}) \\ (\frac{\partial L}{\partial y_{11}} x_{12} + \frac{\partial L}{\partial y_{21}} x_{22}) & (\frac{\partial L}{\partial y_{12}} x_{12} + \frac{\partial L}{\partial y_{22}} x_{22}) & (\frac{\partial L}{\partial y_{13}} x_{12} + \frac{\partial L}{\partial y_{23}} x_{22}) \end{pmatrix}$$

$$= \begin{pmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{pmatrix} \begin{pmatrix} \frac{\partial L}{\partial y_{11}} & \frac{\partial L}{\partial y_{12}} & \frac{\partial L}{\partial y_{13}} \\ \frac{\partial L}{\partial y_{21}} & \frac{\partial L}{\partial y_{22}} & \frac{\partial L}{\partial y_{23}} \end{pmatrix} = X^T \frac{\partial L}{\partial Y}$$

$$\boxed{\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y}}$$

$$\boxed{\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T}$$


dont worry about it if you don't
understand


Don't worry about
it if you don't
understand
- Andrew Ng


me
"If you don't
understand, don't
worry about it"
LINEAR ALGEBRA,
CALCULUS,
PROBABILITY &
STATISTICS...

- http://cs231n.stanford.edu/handouts/linear-backprop.pdf
- http://cs231n.stanford.edu/handouts/derivatives.pdf
- http://cs231n.stanford.edu/slides/2021/lecture_4.pdf
- https://cs231n.github.io/optimization-2/

Model

Sequential API

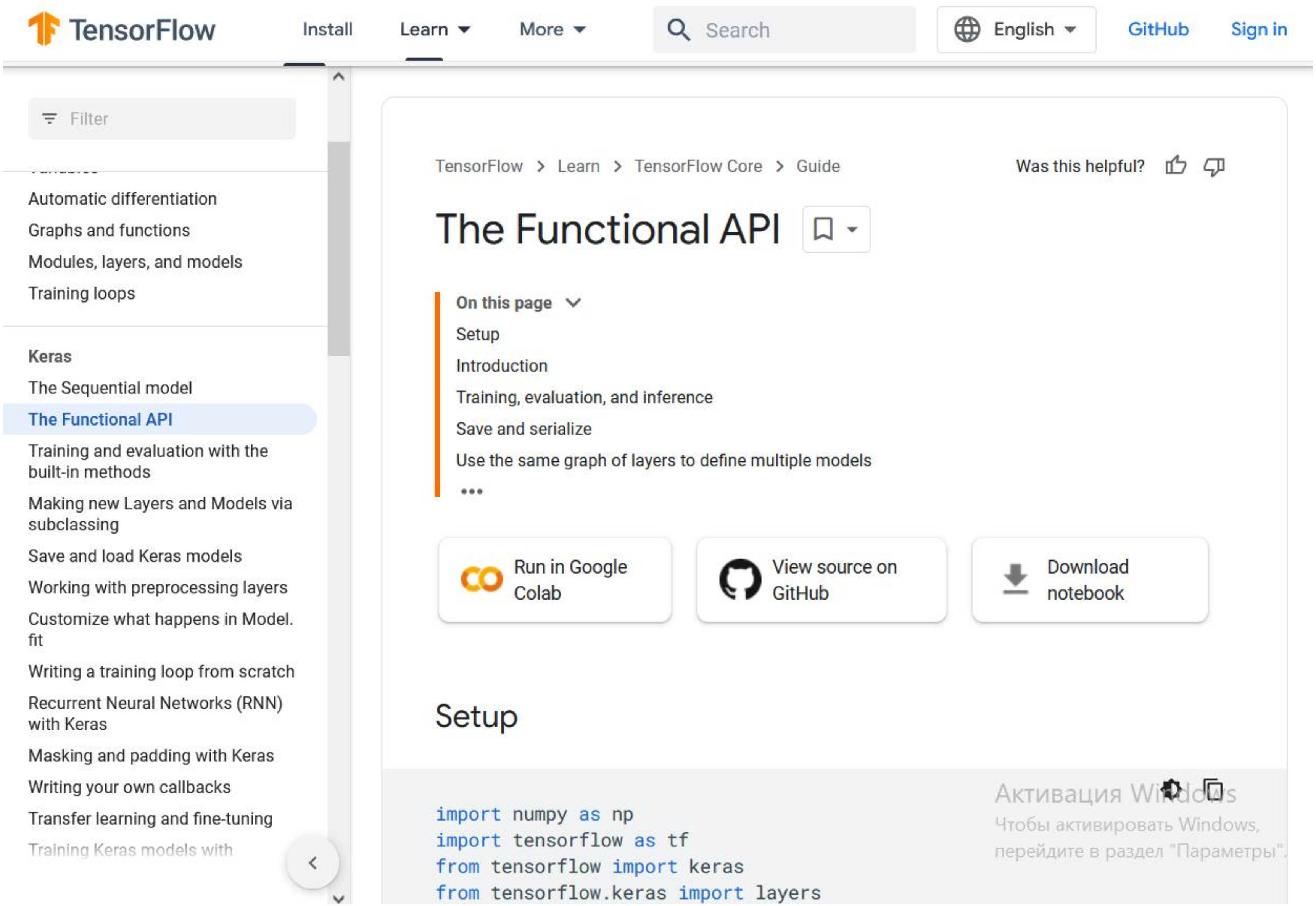
Functional API

Functional model

Input

Layers

Model



TensorFlow | Install | Learn | More | Search | English | GitHub | Sign in

Filter

Automatic differentiation
Graphs and functions
Modules, layers, and models
Training loops

Keras

The Sequential model
The Functional API
Training and evaluation with the built-in methods
Making new Layers and Models via subclassing
Save and load Keras models
Working with preprocessing layers
Customize what happens in Model.fit
Writing a training loop from scratch
Recurrent Neural Networks (RNN) with Keras
Masking and padding with Keras
Writing your own callbacks
Transfer learning and fine-tuning
Training Keras models with

TensorFlow > Learn > TensorFlow Core > Guide

Was this helpful?

The Functional API

On this page

- Setup
- Introduction
- Training, evaluation, and inference
- Save and serialize
- Use the same graph of layers to define multiple models
- ...

Run in Google Colab | View source on GitHub | Download notebook

Setup

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```


https://www.tensorflow.org/guide/keras/functional

```
input = Input(shape=(28,28))
x = Flatten()(input)
x = Dense(128, activation="relu")(x)
predictions = Dense(10, activation="softmax")(x)

func_model = Model(inputs=input, outputs=predictions)
```

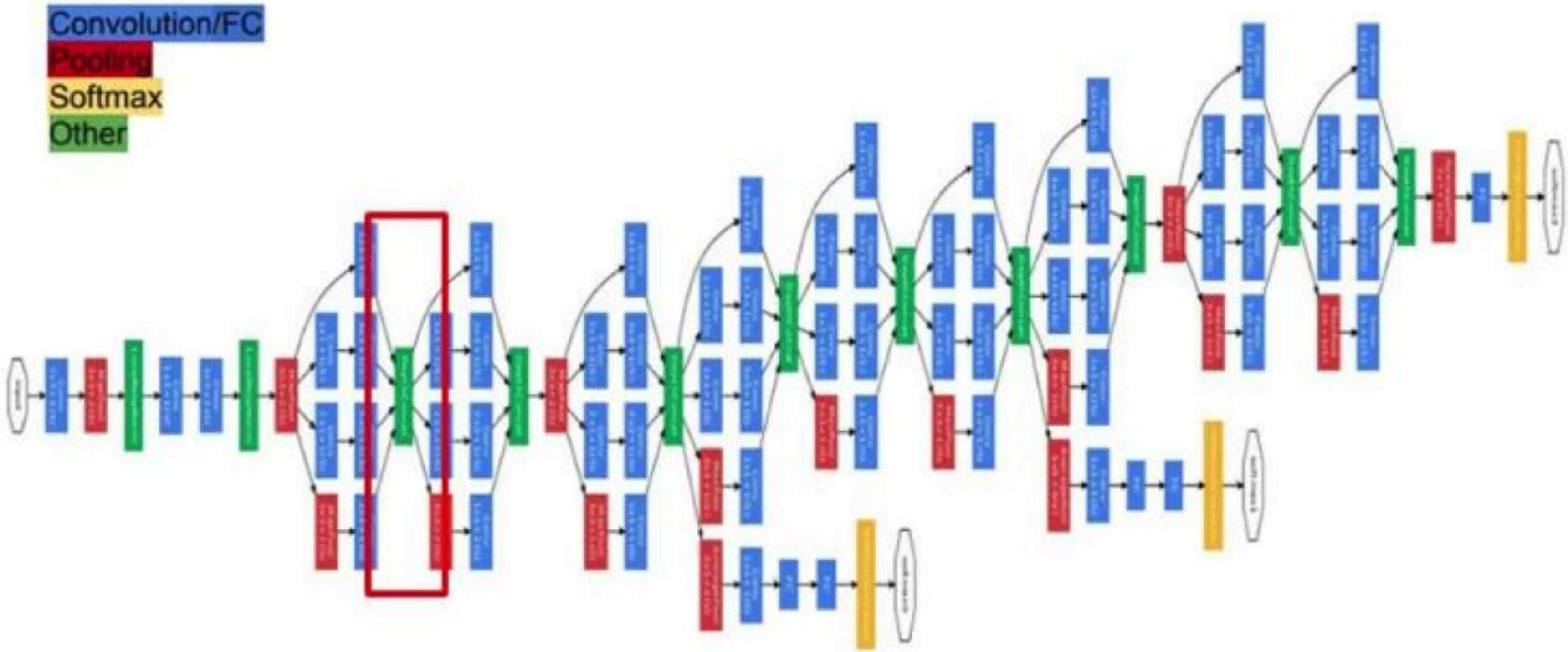
*

```
func_model = Model(inputs=[input1, input2], outputs=[output1, output2])
```

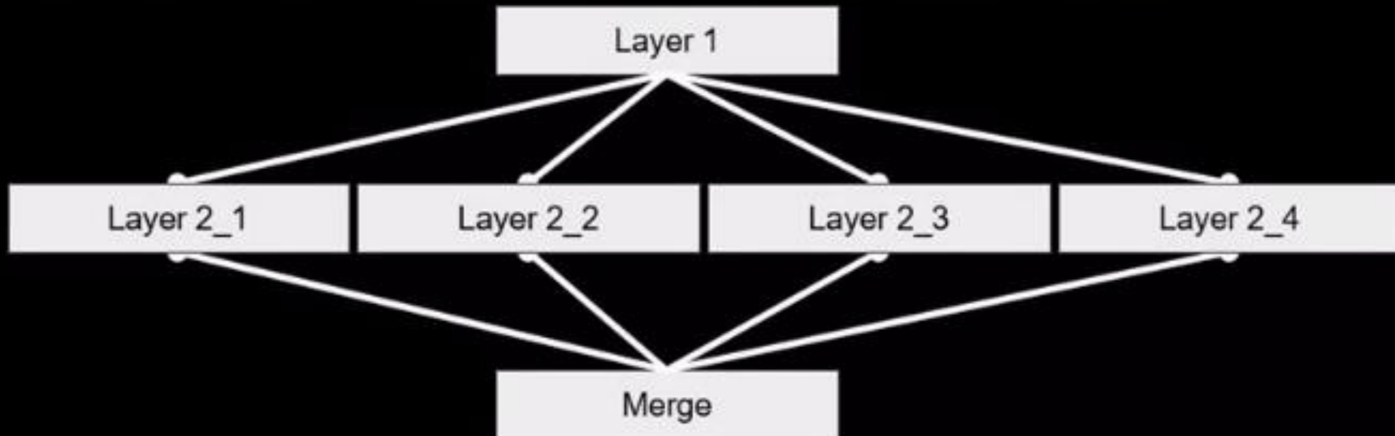
```
dense = layers.Dense(64, activation="relu")
x = dense(inputs)
```

```
model = keras.Model(inputs=inputs, outputs=outputs, name="mnist_model")
```

https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/home/info

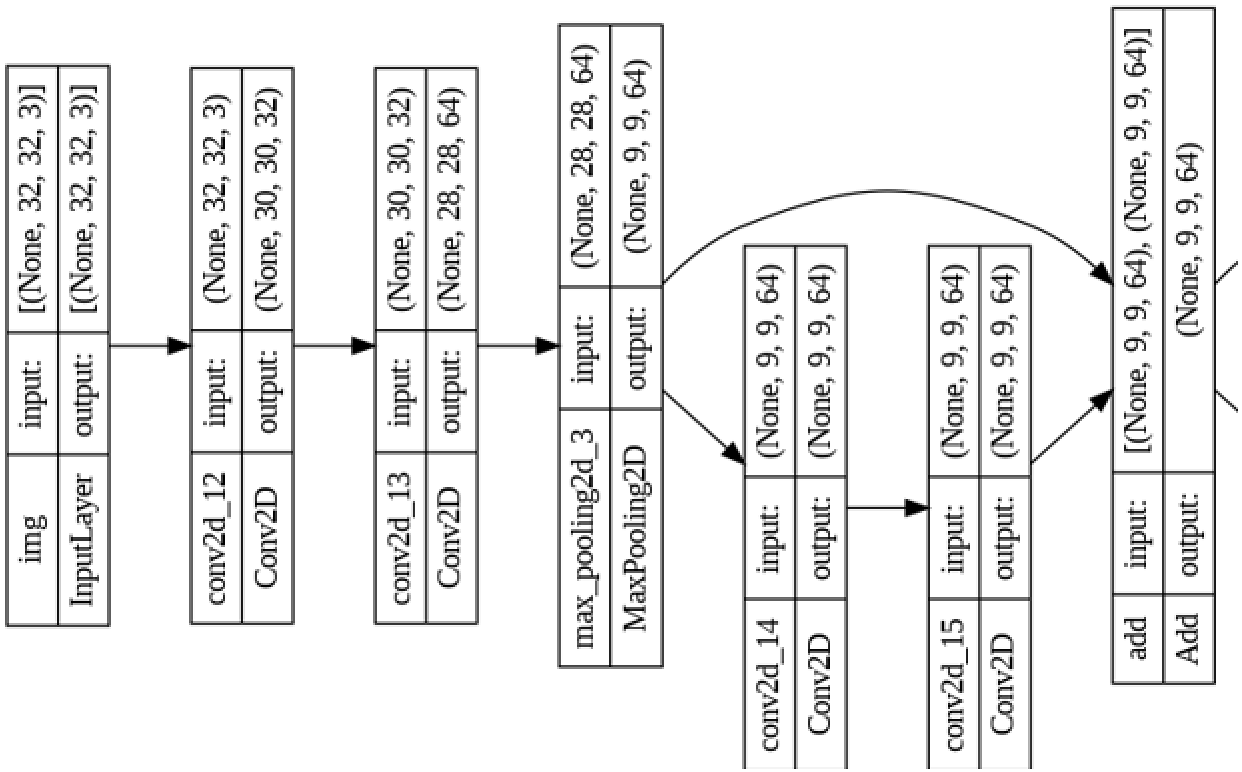

Convolution/FC
Pooling
Softmax
Other

```
layer1 = Dense(32)
layer2_1 = Dense(32)(layer1)
layer2_2 = Dense(32)(layer1)
layer2_3 = Dense(32)(layer1)
layer2_4 = Dense(32)(layer1)
merge = Concatenate([layer2_1, layer2_2, layer2_3, layer2_4])
```



https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/home/info

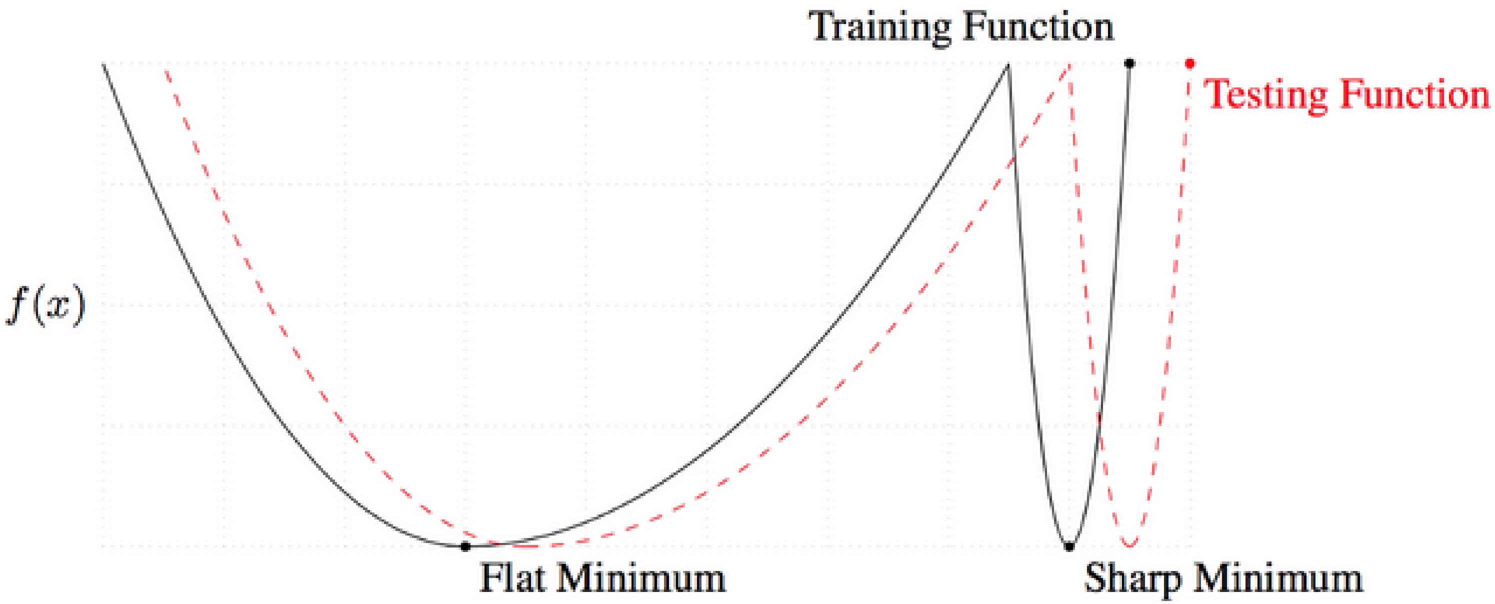
```
inputs = keras.Input(shape=(32, 32, 3), name="img")
x = layers.Conv2D(32, 3, activation="relu")(inputs)
x = layers.Conv2D(64, 3, activation="relu")(x)
block_1_output = layers.MaxPooling2D(3)(x)

x = layers.Conv2D(64, 3, activation="relu", padding="same")(block_1_output)
x = layers.Conv2D(64, 3, activation="relu", padding="same")(x)
block_2_output = layers.add([x, block_1_output])
```



```python
encoder_input = keras.Input(shape=(28, 28, 1), name="img")
x = layers.Conv2D(16, 3, activation="relu")(encoder_input)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(3)(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.Conv2D(16, 3, activation="relu")(x)
encoder_output = layers.GlobalMaxPooling2D()(x)

encoder = keras.Model(encoder_input, encoder_output, name="encoder")
encoder.summary()

x = layers.Reshape((4, 4, 1))(encoder_output)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu")(x)
x = layers.UpSampling2D(3)(x)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
decoder_output = layers.Conv2DTranspose(1, 3, activation="relu")(x)

autoencoder = keras.Model(encoder_input, decoder_output, name="autoencoder")
autoencoder.summary()
```

```python
encoder_input = keras.Input(shape=(28, 28, 1), name="original_img")
x = layers.Conv2D(16, 3, activation="relu")(encoder_input)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(3)(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.Conv2D(16, 3, activation="relu")(x)
encoder_output = layers.GlobalMaxPooling2D()(x)

encoder = keras.Model(encoder_input, encoder_output, name="encoder")
encoder.summary()

decoder_input = keras.Input(shape=(16,), name="encoded_img")
x = layers.Reshape((4, 4, 1))(decoder_input)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu")(x)
x = layers.UpSampling2D(3)(x)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
decoder_output = layers.Conv2DTranspose(1, 3, activation="relu")(x)

decoder = keras.Model(decoder_input, decoder_output, name="decoder")
decoder.summary()

autoencoder_input = keras.Input(shape=(28, 28, 1), name="img")
encoded_img = encoder(autoencoder_input)
decoded_img = decoder(encoded_img)
autoencoder = keras.Model(autoencoder_input, decoded_img, name="autoencoder")
autoencoder.summary()
```

Ансамблі нейронних мереж



Narrow and wide optima. Flat minimum will produce similar loss during training and testing. Narrow loss, however, will give very different results during training and testing. In other words, wide minimum is more generalizable than narrow. Source.

https://towardsdatascience.com/stochastic-weight-averaging-a-new-way-to-get-state-of-the-art-results-in-deep-learning-c639ccf36a

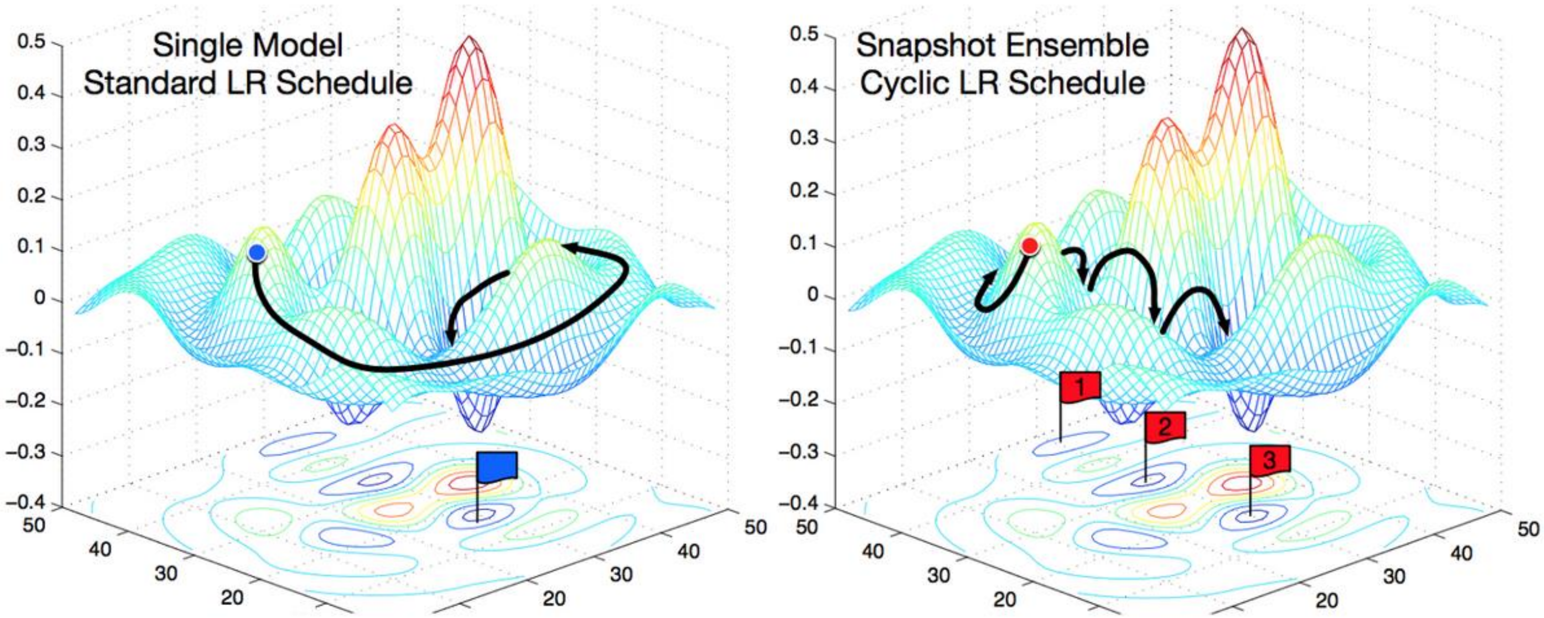
https://arxiv.org/pdf/1609.04836.pdf

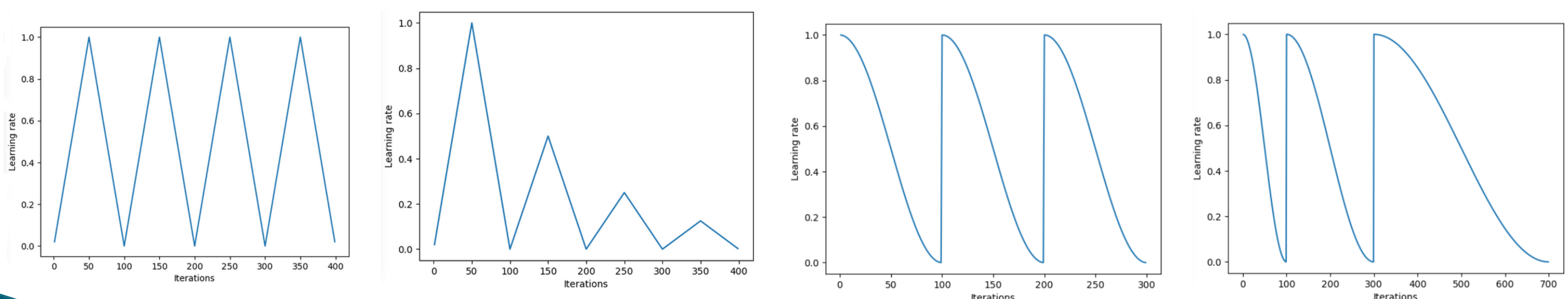
Model Ensembles

- Same model, different initializations
- Top models discovered during cross-validation
- Different checkpoints of a single model
- Running average of parameters during training

$$w_{\text{SWA}} \leftarrow \frac{w_{\text{SWA}} \cdot n_{\text{models}} + w}{n_{\text{models}} + 1},$$

Different checkpoints of a single model



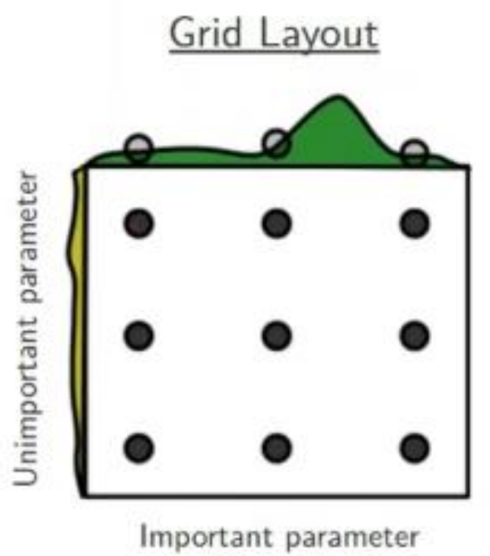


https://arxiv.org/pdf/1704.00109.pdf

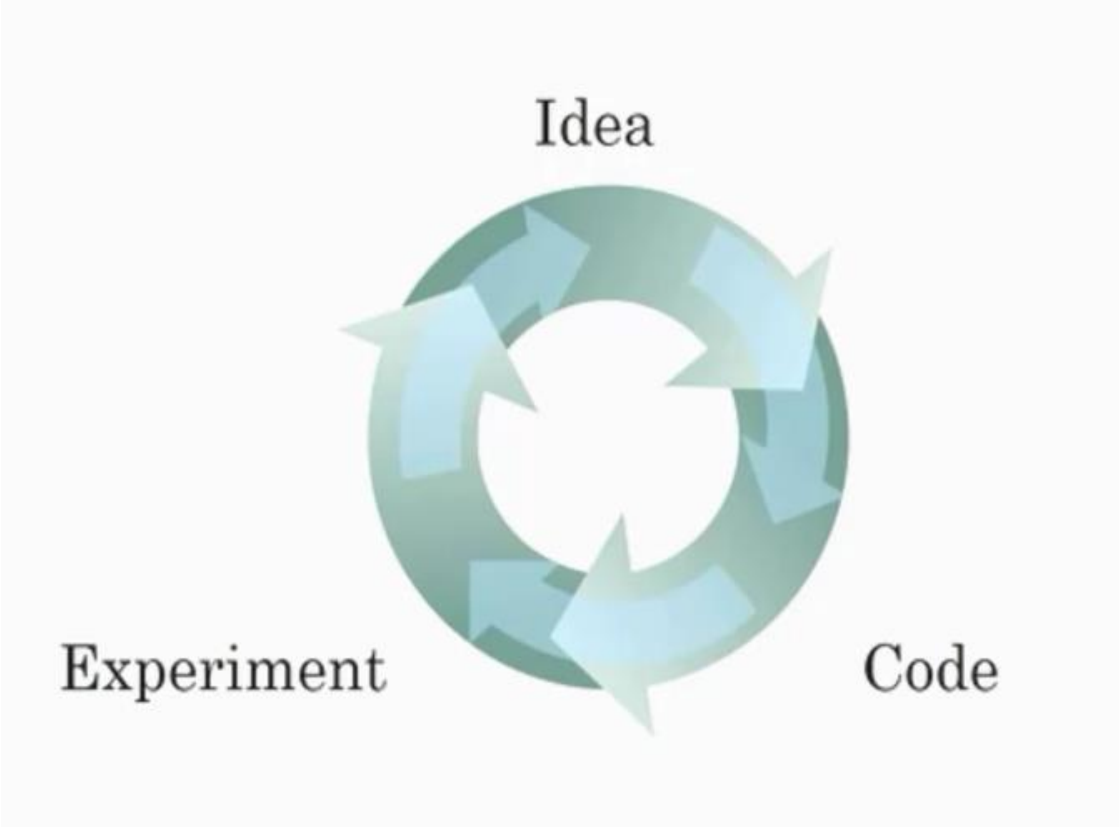
Ensembling

```
inputs = keras.Input(shape=(128,))
y1 = model1(inputs)
y2 = model2(inputs)
y3 = model3(inputs)
outputs = layers.average([y1, y2, y3])
ensemble_model = keras.Model(inputs=inputs, outputs=outputs)
```

Підбір гіперпараметрів



https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf


Idea
Experiment
Code

Introduction to the Keras Tuner

Run in Google Colab | View source on GitHub | Download notebook

Overview

The Keras Tuner is a library that helps you pick the optimal set of hyperparameters for your TensorFlow program. The process of selecting the right set of hyperparameters for your machine learning (ML) application is called *hyperparameter tuning* or *hypertuning*.

Hyperparameters are the variables that govern the training process and the topology of an ML model. These variables remain constant over the training process and directly impact the performance of your ML program. Hyperparameters are of two types:

1. **Model hyperparameters** which influence model selection such as the number and width of hidden layers
2. **Algorithm hyperparameters** which influence the speed and quality of the learning algorithm such as the learning rate for Stochastic Gradient Descent (SGD) and the number of nearest neighbors for a k Nearest Neighbors (KNN) classifier

In this tutorial, you will use the Keras Tuner to perform hypertuning for an image classification application.

https://www.tensorflow.org/tutorials/keras/keras_tuner

K Keras

Search Keras documentation...

Star 57,736

- About Keras
- Getting started
- Developer guides
- Keras API reference
  - Models API
  - Layers API
  - Callbacks API
  - Optimizers
  - Metrics
  - Losses
  - Data loading
  - Built-in small datasets
  - Keras Applications
  - Mixed precision
  - Utilities
  - KerasTuner
    - HyperParameters
    - Tuners

» Keras API reference / KerasTuner / Tuners / Hyperband Tuner

Hyperband Tuner

`Hyperband` class [source]

```
keras_tuner.Hyperband(
    hypermodel=None,
    objective=None,
    max_epochs=100,
    factor=3,
    hyperband_iterations=1,
    seed=None,
    hyperparameters=None,
    tune_new_entries=True,
    allow_new_entries=True,
    max_retries_per_trial=0,
    max_consecutive_failed_trials=3,
    **kwargs
)
```

Variation of HyperBand algorithm.

Reference

Li, Lisha, and Kevin Jamieson. "Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization." Journal of Machine Learning Research 18 (2018): 1-52.

Arguments

- **hypermodel**: Instance of `HyperModel` class (or callable that takes hyperparameters and returns a `Model` instance). It is optional when `Tuner.run_trial()` is overriden and does not use `self.hypermodel`.

https://keras.io/api/keras_tuner/tuners/hyperband/

get_best_hyperparameters method

[source]

```
Tuner.get_best_hyperparameters(num_trials=1)
```

Returns the best hyperparameters, as determined by the objective.

This method can be used to reinstantiate the (untrained) best model found during the search process.

Example

```
best_hp = tuner.get_best_hyperparameters()[0]
model = tuner.hypermodel.build(best_hp)
```

Arguments

- **num_trials**: Optional number of `HyperParameters` objects to return.

Returns

List of `HyperParameter` objects sorted from the best to the worst.

get_best_models method

[source]

```
Tuner.get_best_models(num_models=1)
```

Returns the best model(s), as determined by the tuner's objective.

The base Tuner class

https://keras.io/api/keras_tuner/tuners/base_tuner/

Save and load models

TensorFlow Install Learn More Search English GitHub Sign in

Filter

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
Basic image classification
Basic text classification
Text classification with TF Hub
Regression
Overfit and underfit
Save and load
Tune hyperparameters with the Keras Tuner
More examples on keras.io

Load and preprocess data

ADVANCED

Customization

TensorFlow > Learn > TensorFlow Core > Tutorials

Was this helpful?

Save and load models

On this page

Run in Google Colab | View source on GitHub | Download notebook

Model progress can be saved during and after training. This means a model can resume where it left off and avoid long training times. Saving also means you can share your model and others can recreate your work. When publishing research models and techniques, most machine learning practitioners share:

- code to create the model, and

https://www.tensorflow.org/tutorials/keras/save_and_load