An introduction to AI and **Neural Networks**

@ENSPIMA – Bordeaux INP

Jean-Luc.Charles@ENSAM.EU

September 2022

Welcome to the course "An introduction to AI & Neural Networks"

An introduction to get familiarised with...

- Machine learning
- Training & operating Artificial Neural Networks
- Tensorflow & keras Python modules
- Applications to Predictive Maintenance.

Welcome to the course "An introduction to AI & Neural Networks"

An introduction to get familiarised with...

- Machine learning
- Training & operating Artificial Neural Networks
- Tensorflow & keras Python modules
- Applications to Predictive Maintenance.

Ressources

- 3 hours of lecture and 4 $\times$ practical work Python sessions (4 x 3h) on **your laptop**
- Dedicated github repository with all the course material (PDF, notebooks, videos...)

Practical Work: 4 $\times$ 3h

Self training: Wake up your Python !

- Start with the 2 notebooks `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

Practical Work: 4 $\times$ 3h

Self training: Wake up your Python !

- Start with the 2 notebooks `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

Self training: AI & Machine Learning

- 3 *notebooks* `ML1_MNIST_en.ipynb`, `ML2_DNN_part1_en.ipynb` and `part2` target the skills:
  - load and pre-process MNIST images
  - build a **dense** neural network with tensorflow & keras
  - train the network to recognize MNIST images
  - evaluate and operate the trained network.

Practical Work: 4 $\times$ 3h

Self training: Wake up your Python !

- Start with the 2 notebooks `Wake_up_your_Python-part1.ipynb` and `...part2.ipynb`

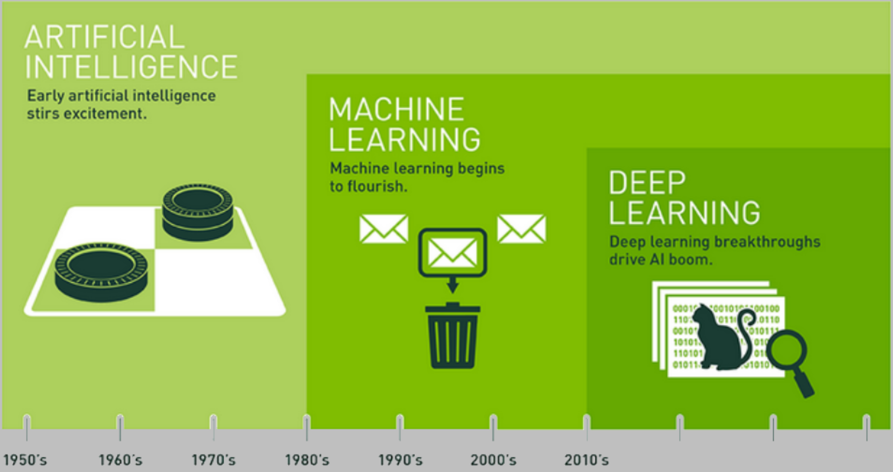
Self training: AI & Machine Learning

- 3 *notebooks* `ML1_MNIST_en.ipynb`, `ML2_DNN_part1_en.ipynb` and `part2` target the skills:
  - load and pre-process MNIST images
  - build a **dense** neural network with tensorflow & keras
  - train the network to recognize MNIST images
  - evaluate and operate the trained network.

Mini-project: application to Predictive Maintenance

- use your skills to process a case of predictive maintenance with a dense neural network...

The historical way...



(from : developer.nvidia.com/deep-learning)

Artificial Intelligence ?

Artificial Intelligence [1]: remains an ambiguous term with multiple definitions varying with time:

- *"...the science of making computers do things that require intelligence when done by humans."* Alan Turing, 1940
- *"the field of study that gives computers the ability to learn without being explicitly programmed."* Arthur Samuel, 1960
- *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* Tom Mitchell, 1997
- Notion of *intelligent agent* or *rational agent*
  "...agent that acts in such a way as to reach the best solution or, in an uncertain environment, the best predictable solution." Stuart Russel, Peter Norvig, "Intelligence Artificielle" 2015

[1] first used in 1956 by John McCarthy, researcher at Stanford during the Dartmouth conference

Artificial Intelligences ?

Strong AI

Weak AI

General AI

Narrow AI

Artificial Intelligences ?

Strong AI

- Build systems that think exactly the same way that people do.
- Try also to explain how humans think...
- Whe are not yet here...

Weak AI

General AI

Narrow AI

Artificial Intelligences ?

Strong AI

- Build systems that think exactly the same way that people do.
- Try also to explain how humans think...
- Whe are not yet here...

Weak AI

- Build systems that can behave like humans.
- The results will tell us nothing about how humans think.
- We already are there... We use it every day!
  (anti-spam, facial or voice recognition, language translation...)

General AI

Narrow AI

Artificial Intelligences ?

Strong AI

- Build systems that think exactly the same way that people do.
- Try also to explain how humans think...
- Whe are not yet here...

Weak AI

- Build systems that can behave like humans.
- The results will tell us nothing about how humans think.
- We already are there... We use it every day!
  (anti-spam, facial or voice recognition, language translation...)

General AI

- AI systems designed for the ability to reason in general.

Narrow AI

- AI systems designed for specific tasks.

Artificial Intelligence

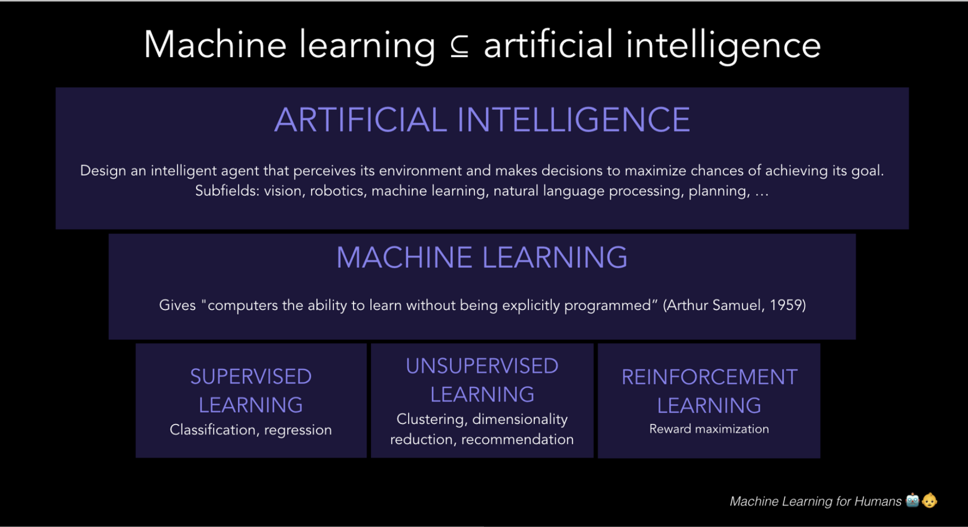
- Runs in much of our present technology (smartphone apps...)
- Powered by rapid advances in data storage, computer processing power
- Powered by free dataset acces via Internet and code publishing as open source environments
- Rate of acceleration is already astounding
- Will likeky shape our future more powerfully than any other innovation this century.

Some AI famous dates

- May 11, 1997: the IBM computer Deep Blue defeated Gary Kasparov at chess.
  Today, Kasparov says: *“computer that defeated me at chess is no more intelligent than an alarm clock”*
  The ability to defeat a human is no more a criteria for defining AI.
- 2011: IBM’s Watson computer wins television game show Jeopardy, defeating legendary human champions.
- 2015: Google Deepmind developped an agent that surpassed human performances at 49 Atari games
- 2016: Google DeepMind’s AlphaGo defeats Go champion Lee Sedol.

Machine Learning and AI

Page from medium.com/machine-learning-for-humans/...



Branches of Machine Learning

Supervised learning

Needs data and labels...

- **Classification**
  - Images classification
  - Objects detection
  - speech recognition...
- **Regression**
  - predict a value...
- **Anomaly detection**
  - Spam detection
  - Manufacturing: finding known (learned) defects
  - Weather prediction
  - Diseases classification...

Branches of Machine Learning

Unsupervised learning

Needs only data...

- **Clustering** – non labelled data **Grouping**
  - Data mining, web data grouping, news grouping...
  - Market segmentation
  - DNA grouping
  - Astronomical data analysis...
- **Anomaly Detection**
  - Fraud detecion
  - Manufacturing: finding defects even new ones
  - Monitoring activity: detecte abnormal activity (failure, hacker, fraud...)
  - Fake account on Internet...
- **Dimensionality reduction**
  - Compress data using fewer numbers...

Branches of Machine Learning

Reinforcement learning

An agent learns to drive an environment...

- **Reward maximisation**
  - ...
- **Control/command**
  - Controlling robots, drones...
  - Factory optimization
  - Financial (stock) trading...
- **Decision making**
  - games (video games)
  - financial analysis...

Machine Learning approaches

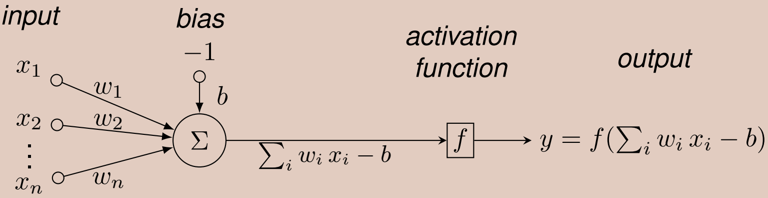
Several approaches/technics can be used to design *Machine Learning* algorithms:

- Genetic programming
- Bayesian inference
- Fuzzy logic
- Neural Networks
- ...

The following deals only with **Artificial Neural Networks**.

The artificial neuron

The computer model of the artificial neuron

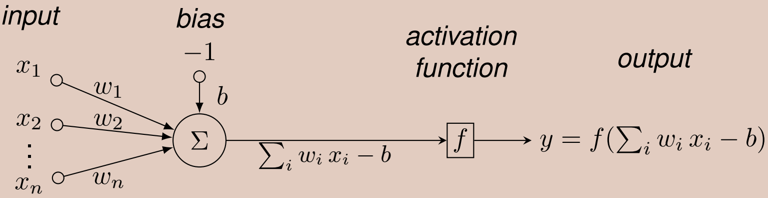


An **artificial neuron**:

- receives the input data $(x_i)_{i=1..n}$ affected by the **weights** $(w_i)_{i=1..n}$ (*weights*)

The artificial neuron

The computer model of the artificial neuron

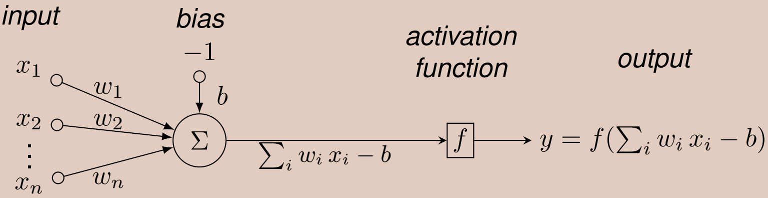


An **artificial neuron**:

- receives the input data $(x_i)_{i=1..n}$ affected by the **weights** $(w_i)_{i=1..n}$ (*weights*)
- calculates the **weighted sum** of its entries minus the bias $\sum_i w_i\, x_i - b$

The artificial neuron

The computer model of the artificial neuron



An **artificial neuron**:

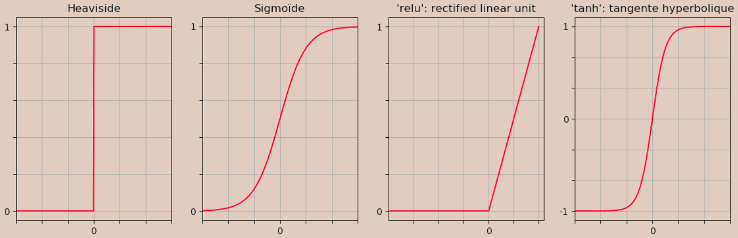
- receives the input data $(x_i)_{i=1..n}$ affected by the **weights** $(w_i)_{i=1..n}$ (*weights*)
- calculates the **weighted sum** of its entries minus the bias $\sum_i w_i\, x_i - b$
- outputs a **activation** $f(\sum_i w_i\, x_i - b)$, computed with an activation function f (generally non-linear).

Artificial neuron

The activation function of a neuron:

- introduces a non-linear behavior,
- sets the range of the neuron output, for example $[-1, 1]$, $[0, 1]$ or even $[0, \infty[$.

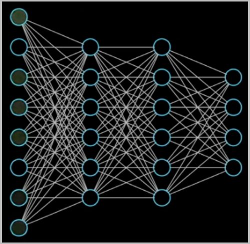
Examples of often used activation functions



- The bias b sets the activation threshold of the neuron.

Neural networks studied

- Neural networks are more or less complex assemblies of artificial neurons grouped by layers.



- Two architectures are very common:
  - The **Dense Neural Network** (DNN), simple, generalist, can perfom greatly when well tuned.
  - The more complex **Convolutional Neural Network** (CNN), mainly specialized in image processing.

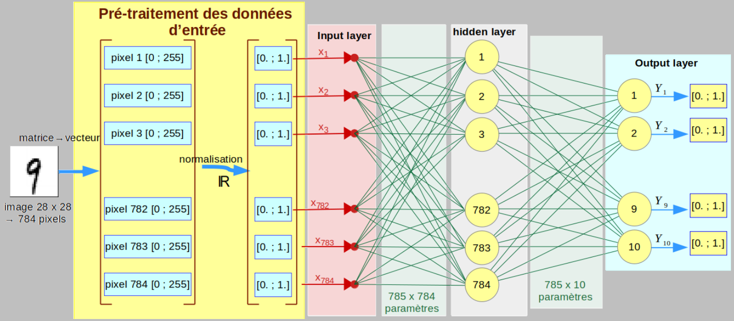
A must example: trainig a Dense Network to classify the MNIST handwritten digit images

- MNIST: bank of 70000 **labeled images** (60000 training images and 10000 test images)

- grayscale images 28 $\times$ 28 pixels.
- Scores with a dense networks can reach 98% success...
- State of the art for image recognition : **Convolutional Neural Networks** (CNN)
  [will not be covered in this course limited to dense networks]

Dense Neural Network architecture

Each matrix 28 $\times$ 28 $\rightsquigarrow$ normalized vector of 784 components `float` $\in [0; 1]$.



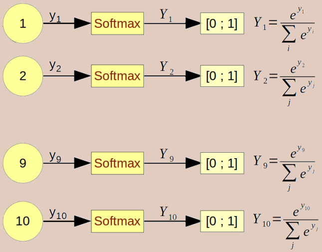
Structure of the network:

- An *Input layer* sets the size of network inputs to 784 values. It has no neurons.
- A *Hidden layer* of 784 neurons (we could have more, or less...), receives the input data. It is connected to the next layer.
- An *Output layer* of 10 neurons (1 neuron for each digit to be recognized).

Activation functions

- In the intermediate layers the activation function *relu* often favors the learning of the network [2] algorithm.
- Classification (last layer) uses the *softmax* function:

Activation function *softmax*



- The activation of neuron k is $Y_k = e^{y_k} / \sum_i e^{y_i}$ with $y_k = \sum_i \omega_i x_i - b$ calculated by the neuron k.
- The outputs of the neurons are interpreted as probabilities in the interval [0,1].

The neuron with the greatest probability (activation) gives the response of the network by its associated label.

[2] avoids the *vanishing gradient* that appears in the *back propagation*

One-hot encoding of labels

Purpose: to put the image labels in the format of the network output

- Image labels: **integers** from 0 to 9.
- Network output: **vector of 10 `float`** in the interval [0,1] calculated by the *softmax* functions of the 10 output neurons.
- *one-hot* coding of an ordered collection of N unique elements:

- each element is coded by a vector of N null components except one,
- the *i th* element $\rightsquigarrow$ vector with a 1 for *i th* component.

chiffre	Y_i' : vecteur *one-hot*
0	[1 0 0 0 0 0 0 0 0 0]
1	[0 1 0 0 0 0 0 0 0 0]
2	[0 0 1 0 0 0 0 0 0 0]
3	[0 0 0 1 0 0 0 0 0 0]
4	[0 0 0 0 1 0 0 0 0 0]
5	[0 0 0 0 0 1 0 0 0 0]
6	[0 0 0 0 0 0 1 0 0 0]
7	[0 0 0 0 0 0 0 1 0 0]
8	[0 0 0 0 0 0 0 0 1 0]
9	[0 0 0 0 0 0 0 0 0 1]

The *one-hot* encoding of labels '0' to '9' results in a 10-component vector, like the one computed by the neural network.

Error function: *Cross entropy error*

- An image processed by the network $\rightsquigarrow$ vector $\hat{Y}$ of 10 `float` to compare to the *hot-one* encoding Y of the label of the image.
- We use the error (or loss) function *cross entropy* adapted to the coding *one-hot*: $e(Y, \hat{Y}) = -\sum_i Y_i\, log(\hat{Y}_i)$

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

actual label one-hot coded

cross entropy error: $-\sum_i Y_i \log(\hat{Y}_i)$

computed probabilities

0	1	2	3	4	5	6	7	8	9
0.1	0.2	0.1	0.3	0.2	0.1	0.9	0.3	0.1	0.2

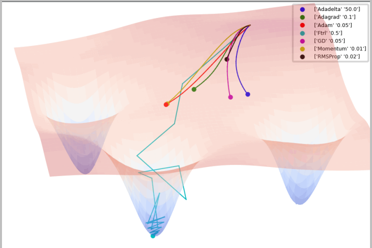
Optimization and *Back Propagation*

- *Feed forward stage* : an optimization algorithm calculates the gradient of the loss function relative to network weights.
- *Back Propagation* : the BP algorithm **modifies** the weights of the network thanks to the gradient of the loss function, iterating from the last layer to the first layer.
- Examples of optimization algorithm used:
  - *Gradient Descent* (GD)
  - *Stochastic Gradient Descent* (SGD)
  - *Adam* (enhanced version of gradient descent)...

  The module tf.keras.optimizers offers Python implementation of several optimization algorithms.

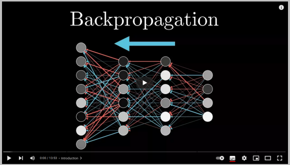
Dense Neural Network

Visualization of gradient descent algorithm iterations for an ultra-simple loss function with only 2 variables:

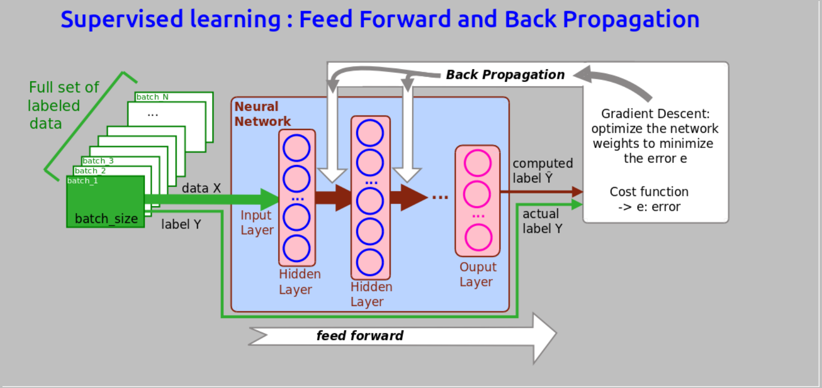


(source: github.com/Jaewan-Yun/optimizer-visualization)

back propagation algorithm explanation video:

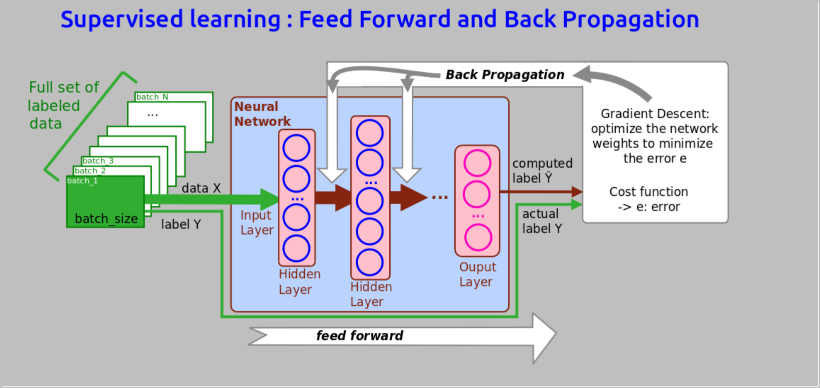


Supervised learning strategy



- The full data set is splitted in (mini) **batches** of size `batch_size`

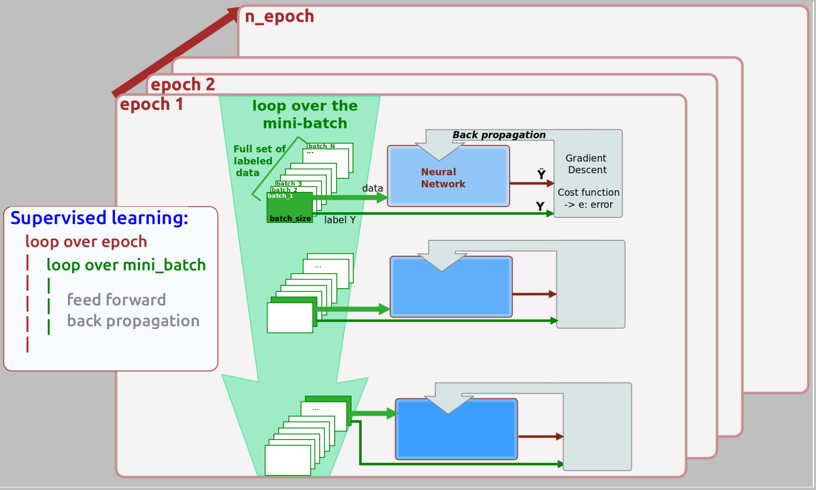
Supervised learning strategy



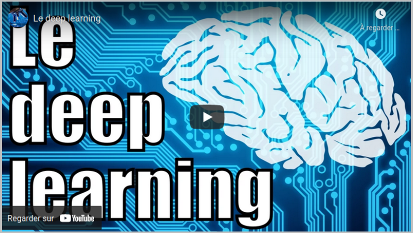
- The full data set is splitted in (mini) **batches** of size `batch_size`
- After each batch has been fed forward: the *Back Propagation* algorithm modifies the weights of the network layer by layer to minimize the error e.

Supervised learning strategy

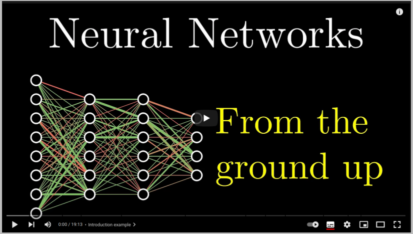
- The training over the ful data set is repeated `n_epoch` times....



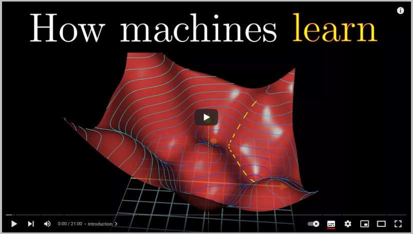
Videos



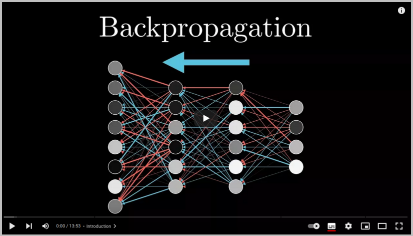
1/ Local: "Le deep learning - YouTube.webm"



2/ local : "But what is a neural network.webm"



3/ Local: "Gradient descent how neural networks learn.webm"



4/ Local: "What is backpropagation really doing .webm"

References

[1] *Artificial Intelligence: A Modern Approach (4th Edition)*, By Stuart Russell & Peter Norvig. Pearson, 2020. ISBN 978-0134610993. aima.cs.berkeley.edu

Intelligence artificielle – Une approche moderne – 4e éd., By Stuart Russell & Peter Norvig. Translated by L. Miclet, F. Popineau, & C. Cadet. Paris: Pearson Education France, 2021. ISBN 978-2326002210.

[2] *What is artificial intelligence (AI), and what is the difference between general AI and narrow AI?*, Kris Hammond, 2015 www.computerworld.com/article/2906336/what-is-artificial-intelligence.html

[3] *Stanford Encyclopedia of Philosophy*, plato.stanford.edu/entries/artificial-intelligence

[4] *Deep Learning.*, Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016), MIT Pres, ISBN 9780262035613