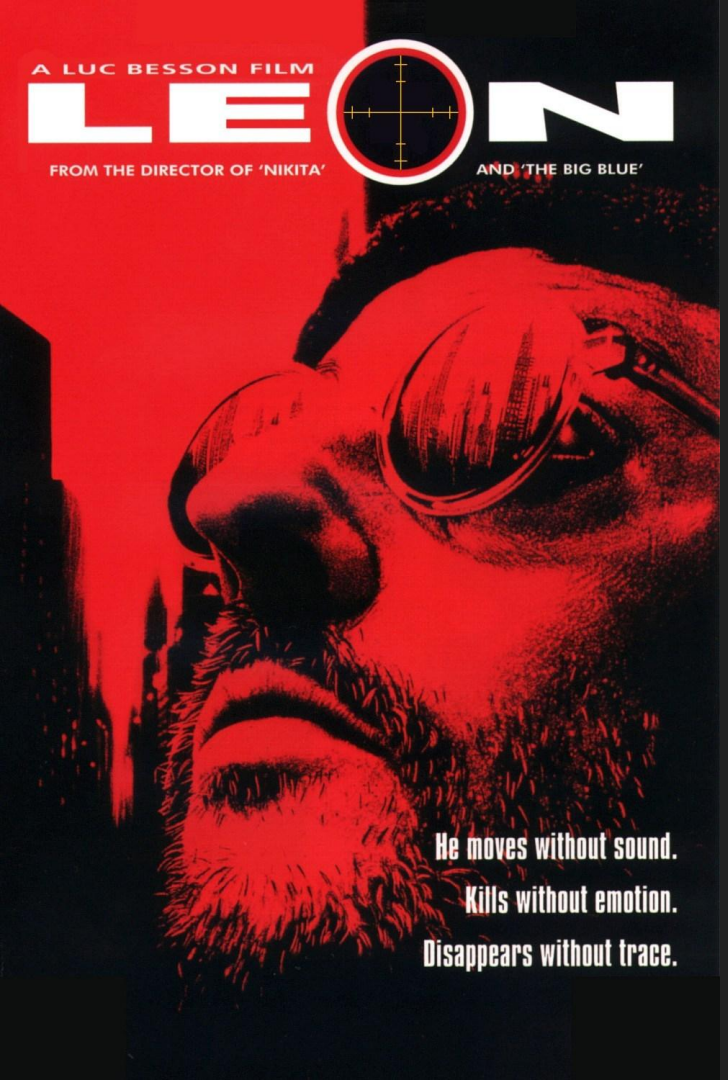


MARIANA

The **Cutest**
Deep Learning
Framework

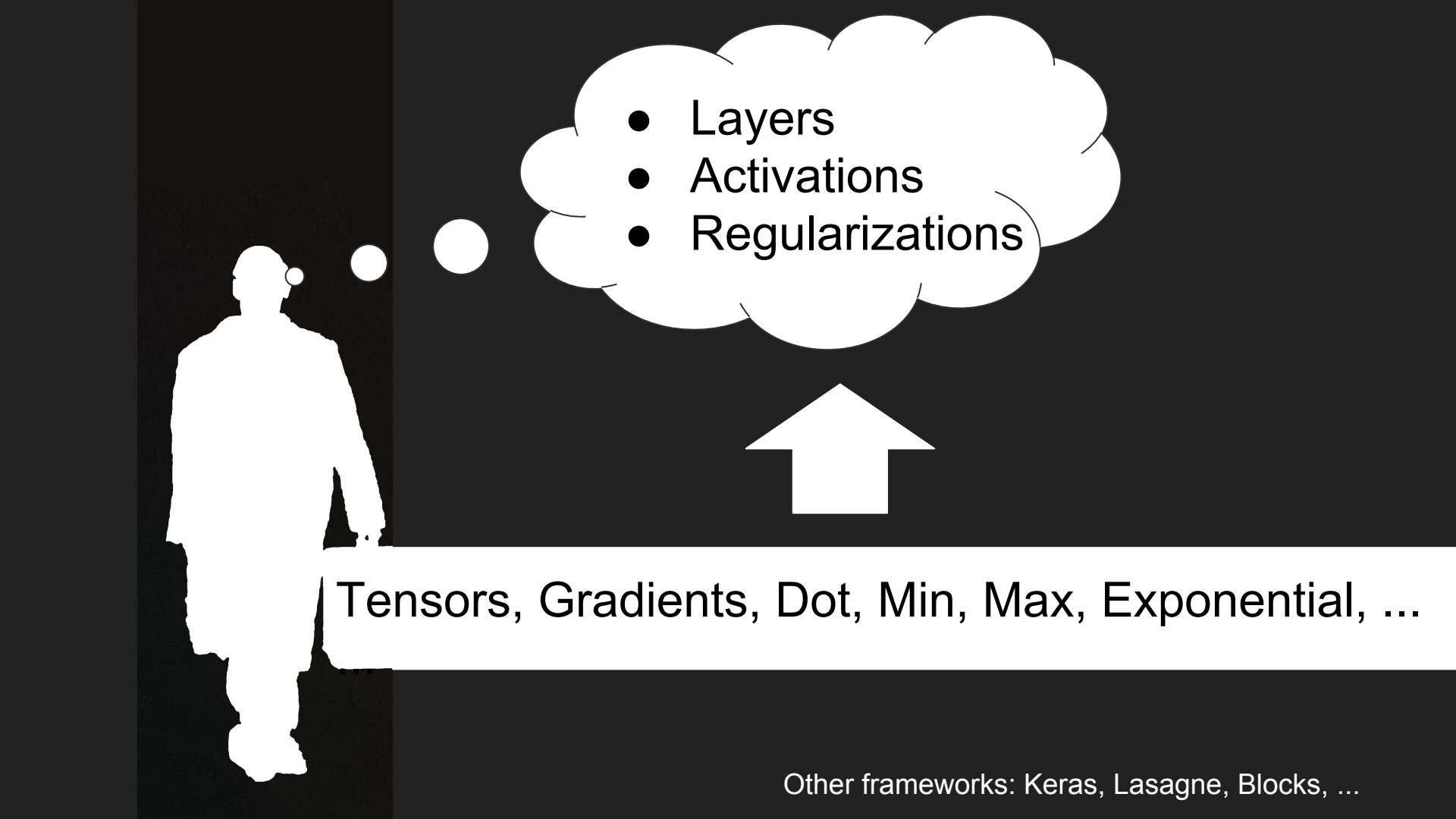
Deep Learning is Hot

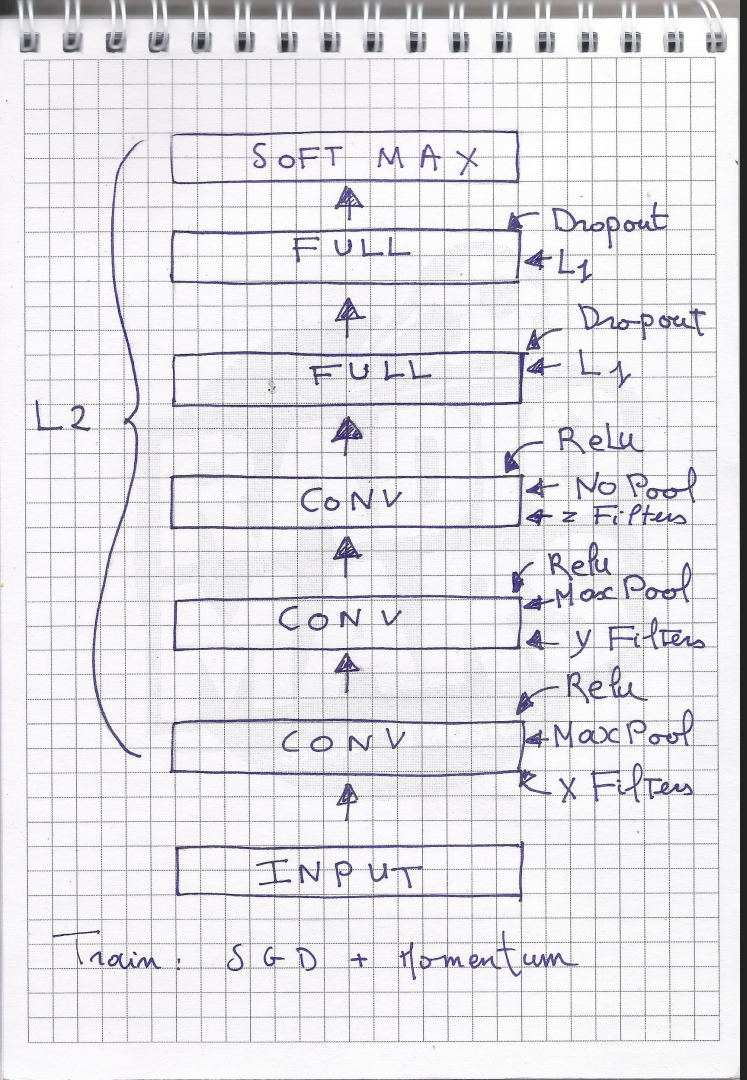
- Deep Learning is getting out of the labs
- Big companies are interested
- There’s is more ML being done
- More people want to learn it

- Time consuming
- Repetitive
- Hard to debug
- Hard to teach

- Very Hard to learn

Neural Nets Complexity


Layers
Activations
Regularizations
Tensors, Gradients, Dot, Min, Max, Exponential, ...
Other frameworks: Keras, Lasagne, Blocks, ...



User extensions

Language

(MARIANA)

CODE

Theano

- Built-in optimizations
- GPU
- Python

MARIANA

- Feedforward
  - Fully connected
  - Convolution
  - Embeddings
- No restriction in connectivity
  - Multi-Inputs, Multi-Outputs, Forks
- Layer independent hyper-parameters
- Save / Reload models
- Export to HTML for visualization
- Fully Documented
- No recurrent nets yet but it is planned
- Github

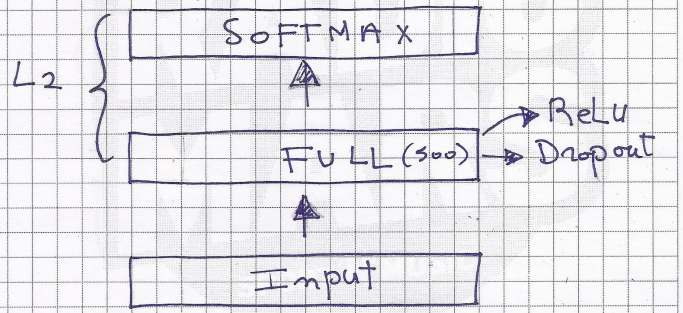
MARIANA abstractions

- Layers
- Learning Scenarios
  - Gradient Descent, Momentum, ...
- Costs
  - Negative log Likelihood, Mean Squared Error, …
- Regularizations
  - L1, L2, ...
- Decorators
  - Weight initialisations, Dropout, ...

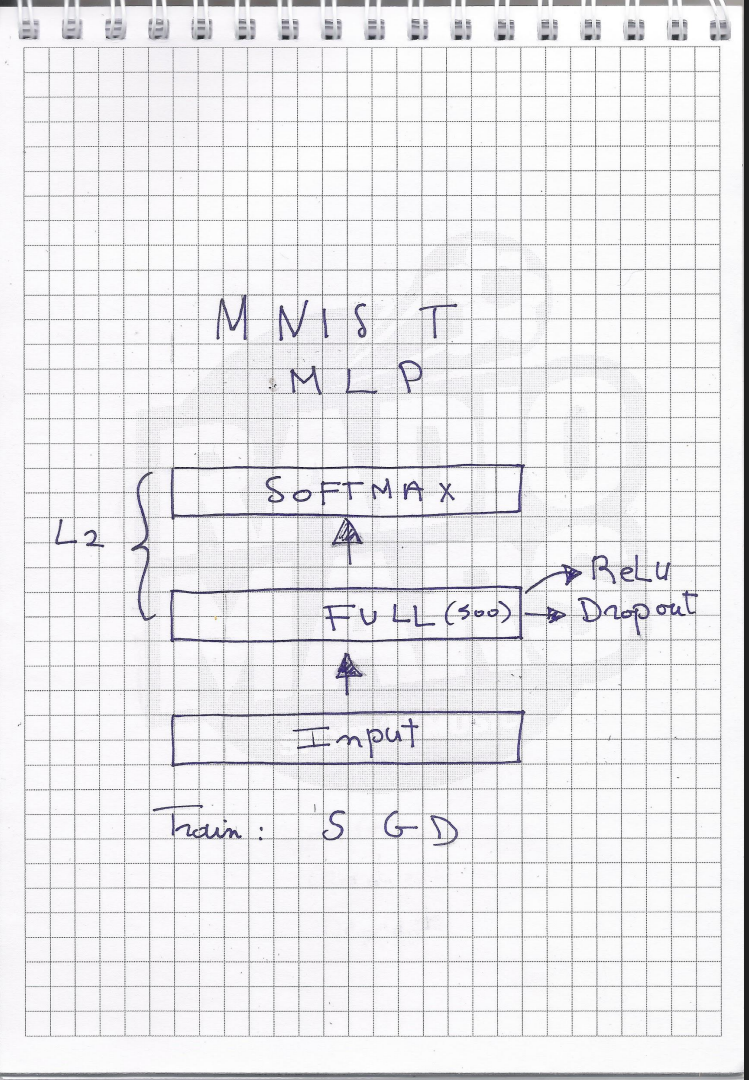
MARIANA: Things that I may not cover

- Trainer (encapsulate all the training)
  - Stop criteria (Early stopping, ...)
  - Dataset mappers (Oversampling, ...)
  - Recorders (record / print)
  - **Emergency savings**

MNIST MLP



Train: SGD



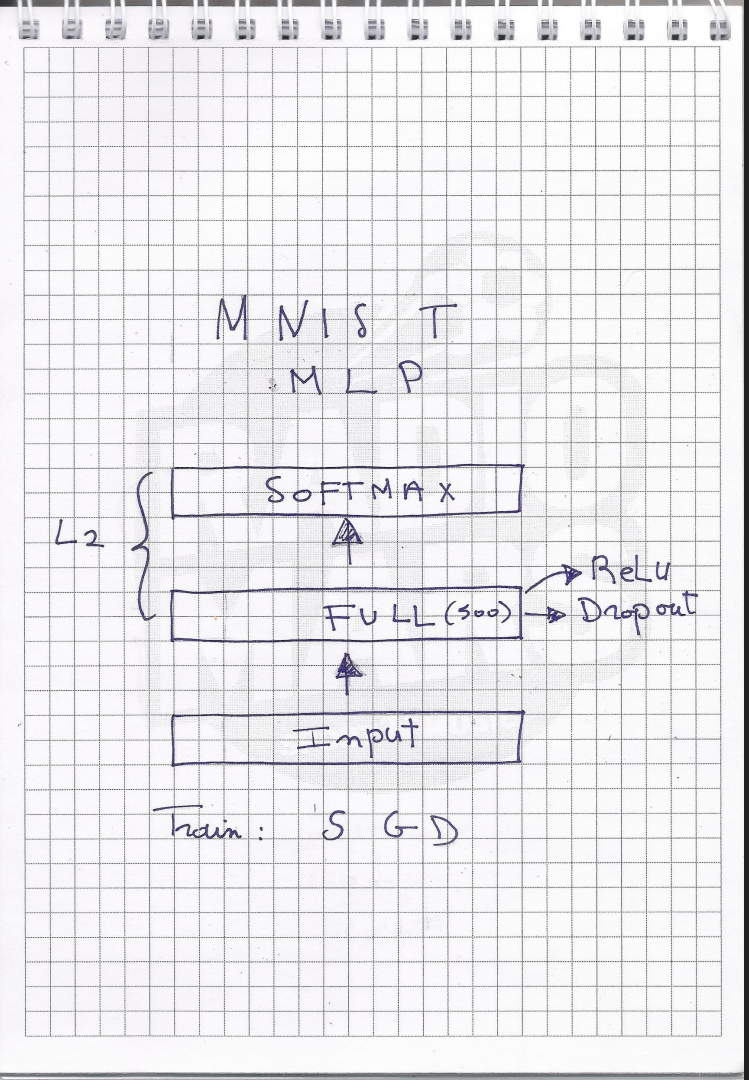

```
ls = MS.GradientDescent(lr = 0.01)
cost = MC.NegativeLogLikelihood()

i = ML.Input(28*28, name = 'InputLayer')

h = ML.Hidden(500,
    activation = MA.ReLU(),
    decorators = [MD.BinomialDropout(0.1)],
    regularizations = [ MR.L2(0.0001) ],
    name = "Hidden" )

o = ML.SoftmaxClassifier(10,
    learningScenario = ls,
    costObject = cost,
    regularizations = [ MR.L2(0.0001) ],
    name = "OutputLayer")

mlp = i > h > o
```




```python
train_set, validation_set, validation_set = load_mnist()

miniBatchSize = 20
for epoch in xrange(100) :
    for i in xrange(0, len(train_set[0]), miniBatchSize) :
    mlp.train("OutputLayer",
        InputLayer = train_set[0][i : i +miniBatchSize],
        targets = train_set[1][i : i +miniBatchSize] )

    mlp.test("OutputLayer",
        inp = validation_set[0],
        targets = validation_set[1] )
```

Special Thanks

- Testers:
  - Jonathan Séguin (IRIC)
  - Assya Trofimov (IRIC)
- Theano devs:
  - Frédéric Bastien (MILA, UdeM)
  - Pascal Lamblin (MILA, UdeM)
- Supervisors:
  - Claude Perreault (IRIC)
  - Sébastien Lemieux (IRIC)