Model training

Roberto Doriguzzi Corin

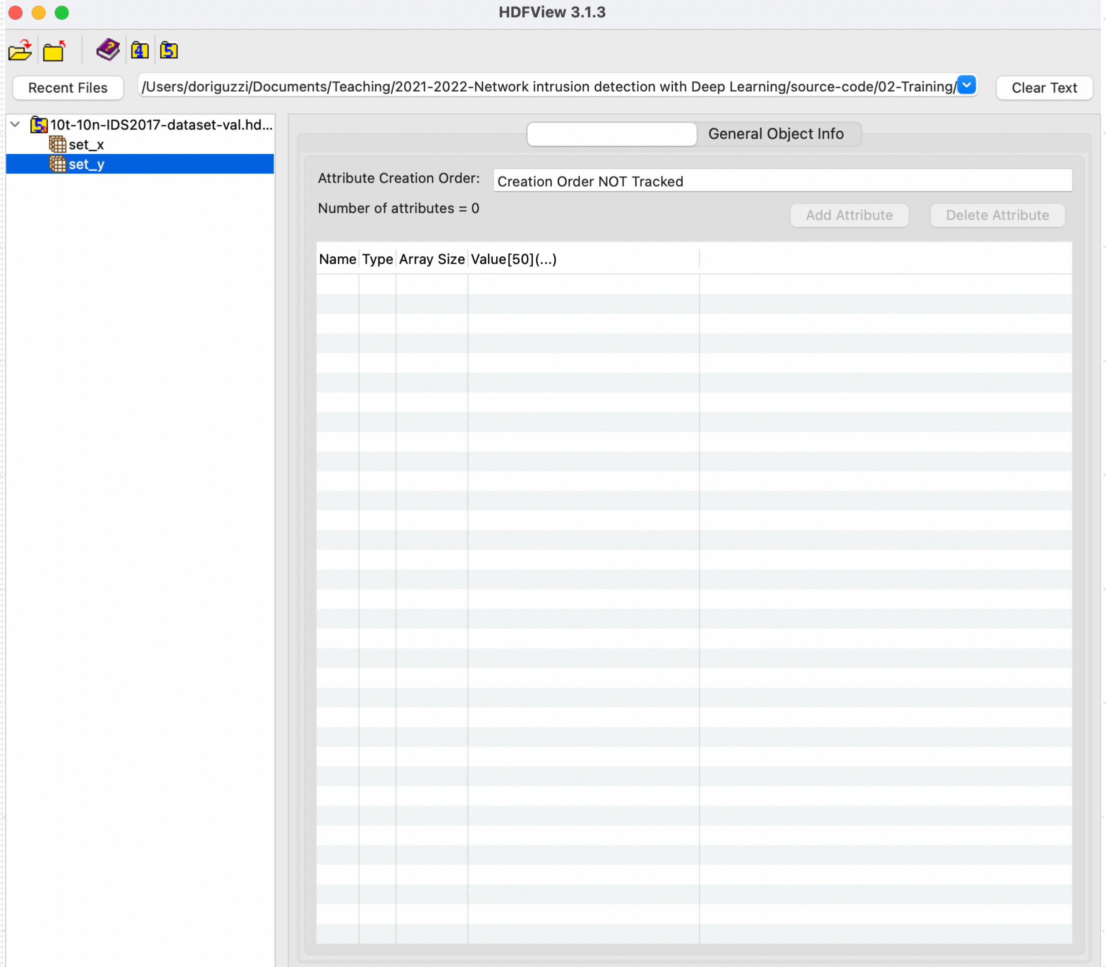
Cybersecurity centre

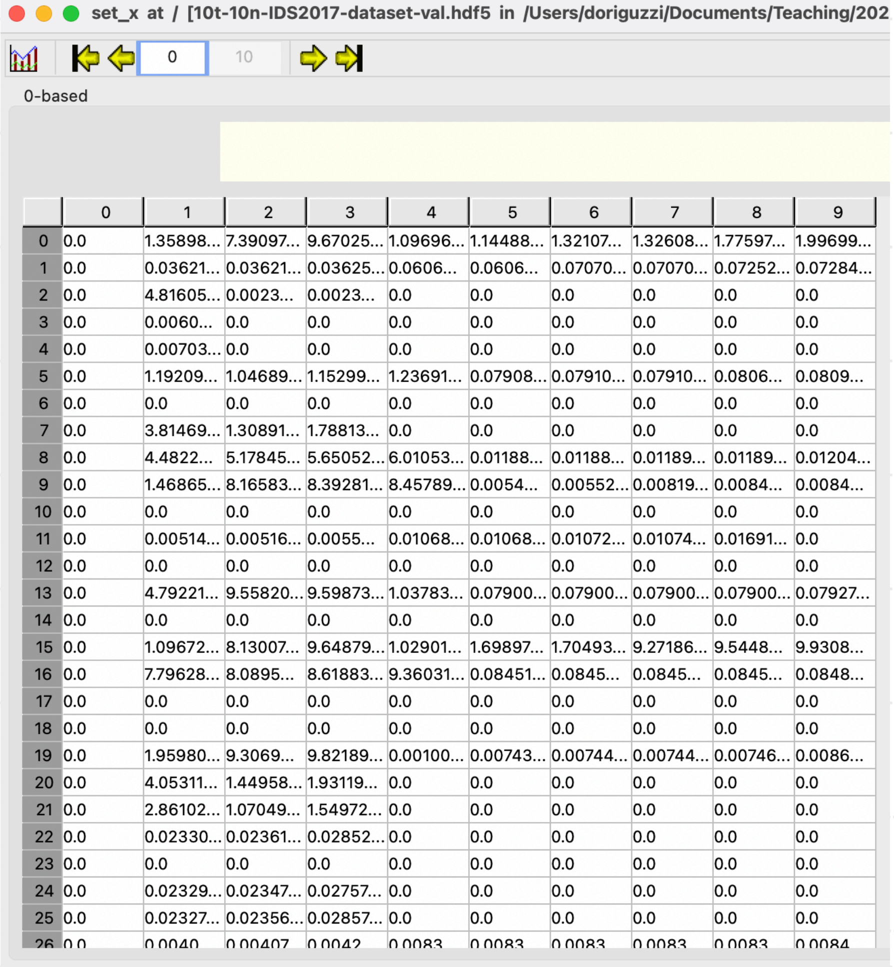
Fondazione Bruno Kessler

Outline

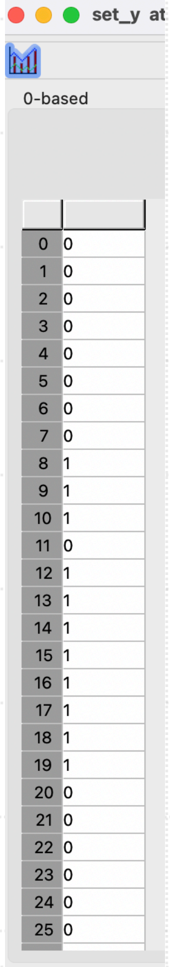
- Hdf5 file format
- Optimizers
- Training a binary model for DDoS attack detection
- Laboratory: implement a multi-class model for DDoS attack detection

Hdf5 file format





	0	1	2	3	4	5	6	7	8	9
0	0.0	1.35898...	7.39097...	9.67025...	1.09696...	1.14488...	1.32107...	1.32608...	1.77597...	1.99699...
1	0.0	0.03621...	0.03621...	0.03625...	0.0606...	0.0606...	0.07070...	0.07070...	0.07252...	0.07284...
2	0.0	4.81605...	0.0023...	0.0023...	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0060...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.00703...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	1.19209...	1.04689...	1.15299...	1.23691...	0.07908...	0.07910...	0.07910...	0.0806...	0.0809...
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	3.81469...	1.30891...	1.78813...	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	4.4822...	5.17845...	5.65052...	6.01053...	0.01188...	0.01188...	0.01189...	0.01189...	0.01204...
9	0.0	1.46865...	8.16583...	8.39281...	8.45789...	0.0054...	0.00552...	0.00819...	0.0084...	0.0084...
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.00514...	0.00516...	0.0055...	0.01068...	0.01068...	0.01072...	0.01074...	0.01691...	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	4.79221...	9.55820...	9.59873...	1.03783...	0.07900...	0.07900...	0.07900...	0.07900...	0.07927...
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	1.09672...	8.13007...	9.64879...	1.02901...	1.69897...	1.70493...	9.27186...	9.5448...	9.9308...
16	0.0	7.79628...	8.0895...	8.61883...	9.36031...	0.08451...	0.0845...	0.0845...	0.0845...	0.0848...
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	1.95980...	9.3069...	9.82189...	0.00100...	0.00743...	0.00744...	0.00744...	0.00746...	0.0086...
20	0.0	4.05311...	1.44958...	1.93119...	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	2.86102...	1.07049...	1.54972...	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.02330...	0.02361...	0.02852...	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.02329...	0.02347...	0.02757...	0.0	0.0	0.0	0.0	0.0	0.0
25	0.0	0.02327...	0.02356...	0.02857...	0.0	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0040	0.00407	0.0042	0.0083	0.0083	0.0083	0.0083	0.0083	0.0084



0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	1
9	1
10	1
11	0
12	1
13	1
14	1
15	1
16	1
17	1
18	1
19	1
20	0
21	0
22	0
23	0
24	0
25	0

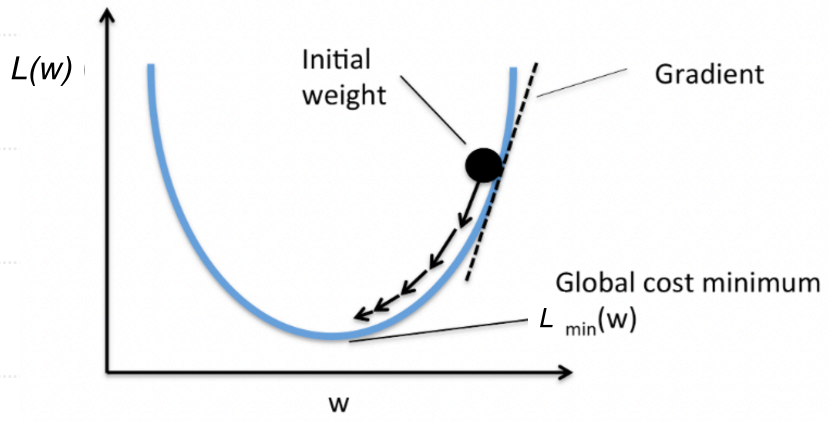
Optimizers

Optimizers *are algorithms or methods used to change the parameters of your neural network (e.g., weights and learning rate) in order to reduce the loss.*

Learning Rate *defines how big/small the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate.*

Gradient Descent

- **Batch gradient descent:** all the data used into a single step per training epoch. We take the average of the gradients of all the training samples. Vectorization can be used to leverage parallel computation.
- **Stochastic gradient descent:** one sample for each step. The parameters are updated more frequently.
- **Mini-batch gradient descent:** steps with mini-batches of samples. Multiple steps each epoch, with the gain in speed thanks to vectorization.

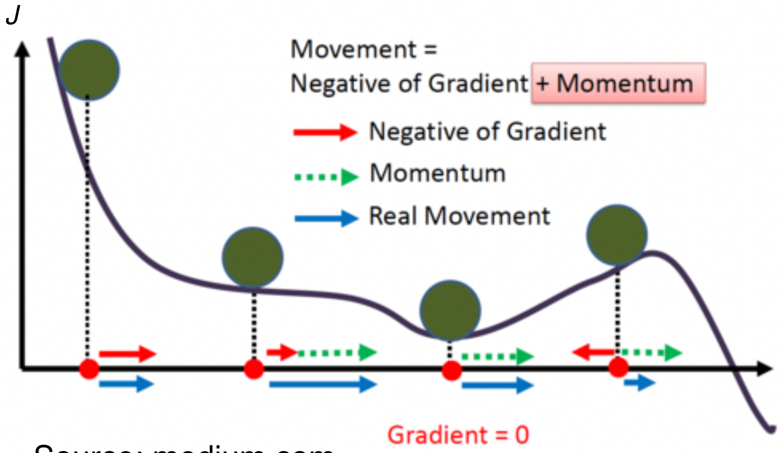


Source: sebastianraschka.com

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-I}}$$

Gradient Descent with Momentum

- Momentum is a hyper-parameter (like the learning rate) that has been introduced to:
  - Faster escape from plateaus
  - Escape from a local mimima
  - Tune the updates based on past gradients
  - Momentum $\beta = 0.9$ works well in practice



Source: medium.com

Learning rate

$$w_t = w_{t-1} - \alpha \cdot m_t$$

$$m_t = \beta m_{t-1} + (1-\beta)\frac{\partial J}{\partial w_{t-1}}$$

Exponential moving average

Adam (adaptive momentum estimation)

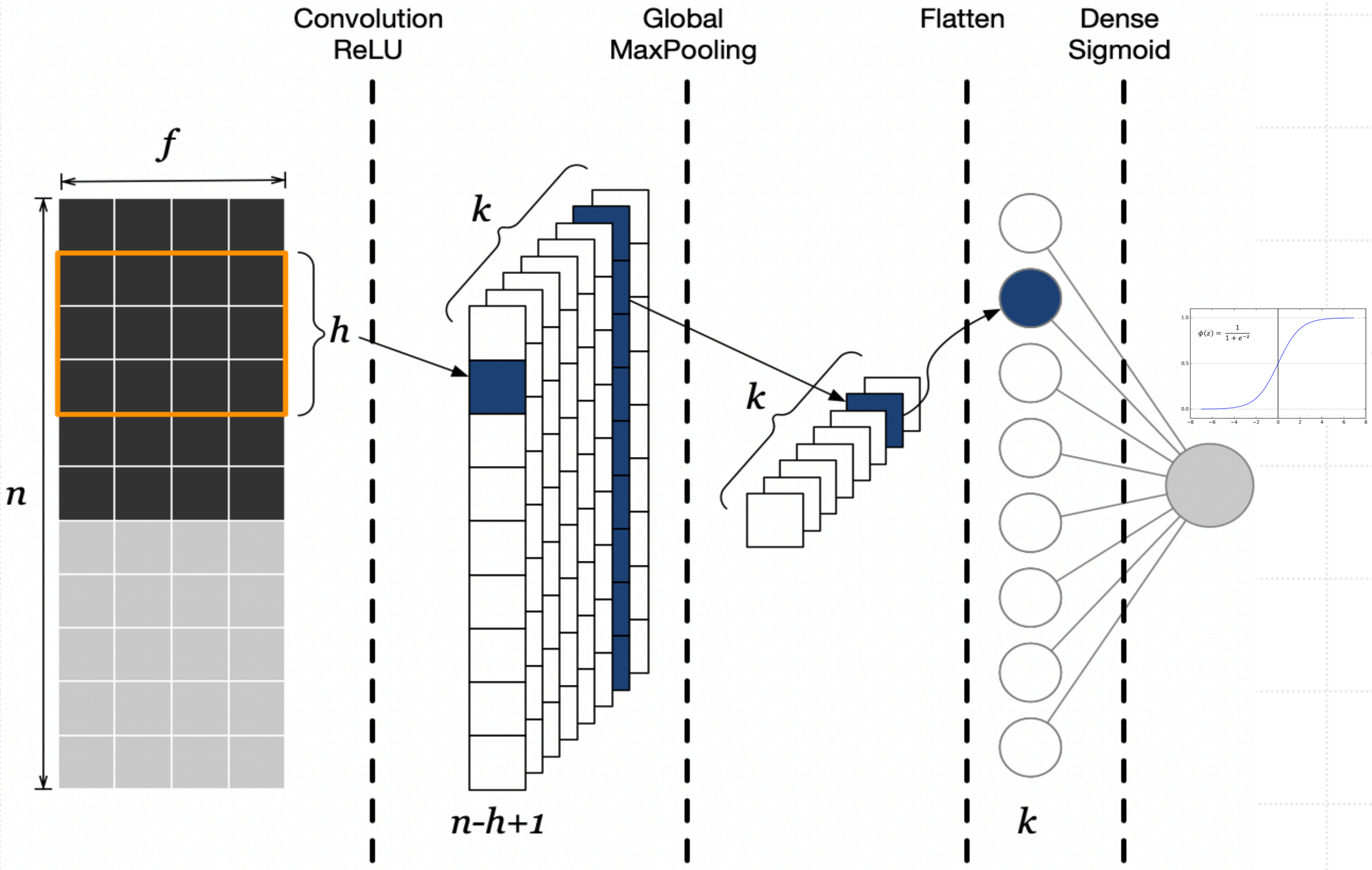
- Adam is an optimisation algorithm that can be used instead of the classical stochastic gradient descent
- The algorithm calculates an **exponential moving average** of the gradient and the squared gradient, and the parameters β_1 and β_2 control the decay rates of these moving averages (usually set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$).
- Converges faster than SGD on large problems in terms of data and parameters

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \frac{\partial J}{\partial w_{t-1}} \qquad v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \left[\frac{\partial J}{\partial w_{t-1}}\right]^2$$

Training a binary classifier

- CNN model
- Pre-processed balanced dataset of benign and DDoS attack traffic (HTTP attack from the CIC-IDS2017 dataset, generated with the LOIC tool https://www.imperva.com/learn/ddos/low-orbit-ion-cannon/)
- Make it run and play with the hyper-parameters
- Change the optimizer
- Change the model (e.g., replace the CNN with a MLP)

Architecture of the CNN model



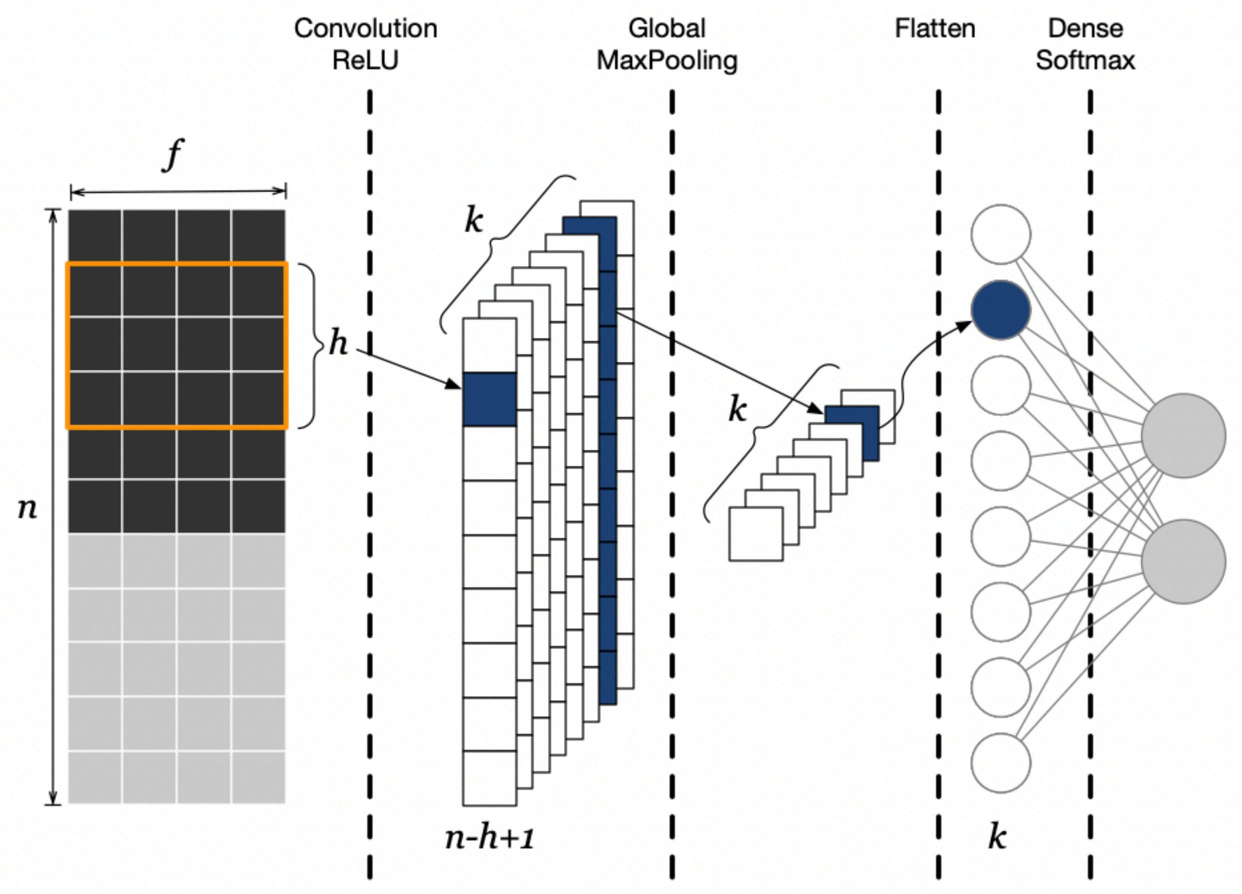
Hyperparameters

- f=11
- n=10
- k=32
- h=3
- Learning rate = 0.01

Lab: implement a 2-class classifier

- Same CNN model with different output
- Replace the binary classifies with a 2-class classifier
- Pay attention to the format of the labels

Architecture of the multi-class CNN model



Hyperparameters

- *f*=11
- *n*=10
- *k*=32
- *h*=3
- Learning rate = 0.01