UNIVERSITY

LEVEL's Degree in COURSE

POLITECNICO DI TORINO
1859 1906

LEVEL's Degree Thesis

Study and development of fault tolerant operating systems for aerospace applications

Supervisors

Prof. Luca STERPONE

Prof. NAME SURNAME

Prof. NAME SURNAME

Candidate

Salvatore Gabriele LA GRECA

MONTH YEAR

Summary

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Acknowledgements

ACKNOWLEDGMENTS

"HI"
Goofy, Google by Google

Table of Contents

List of Tables

List of Figures

Acronyms

AI

artificial intelligence

Chapter 1

Hello

[Hi 1, Goofy]
$\mathrm{kg\,s^{-1}}$

Figure 1.1: Hi

1.1 Extremely long name with manual linebreak which otherwise would not fit the page

1. A
2. B
3. C

Figure 1.2: HI

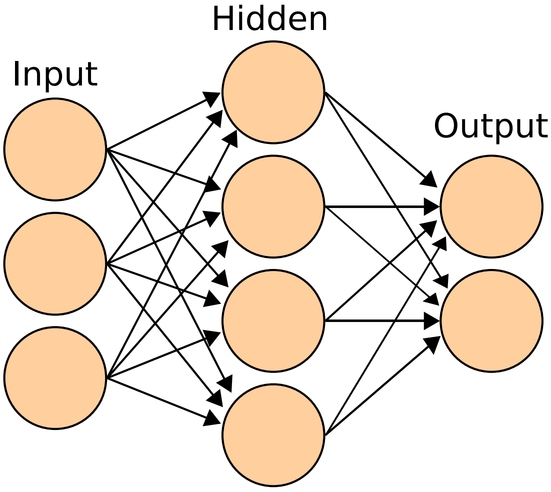

Figure 1.3: SVG

ReLU	$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$
Softmax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} i = 1, ..., J$
tanh	$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$

Table 1.1: Examples of activation functions, operating either element-wise or vector-wise, depending on the function

$$output = f_{activation}\left(\sum_{\#neurons} input_i + bias\right) \tag{1.1}$$

- A
- B
- C

Algorithm 1 Adam optimizer algorithm. All operations are element-wise, even powers. Good values for the constants are $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. ϵ is needed to guarantee numerical stability.

```
procedure ADAM($\alpha, \beta_1, \beta_2, f, \theta_0$)
    ▷ $\alpha$ is the stepsize
    ▷ $\beta_1, \beta_2 \in [0, 1)$ are the exponential decay rates for the moment estimates
    ▷ $f(\theta)$ is the objective function to optimize
    ▷ $\theta_0$ is the initial vector of parameters which will be optimized
    ▷ Initialization
    $m_0 \leftarrow 0$        ▷ First moment estimate vector set to 0
    $v_0 \leftarrow 0$        ▷ Second moment estimate vector set to 0
    $t \leftarrow 0$        ▷ Timestep set to 0
    ▷ Execution
    while $\theta_t$ not converged do
        $t \leftarrow t + 1$        ▷ Update timestep
        ▷ Gradients are computed w.r.t the parameters to optimize
        ▷ using the value of the objective function
        ▷ at the previous timestep
        $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
        ▷ Update of first-moment and second-moment estimates using
        ▷ previous value and new gradients, biased
        $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
        $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
        ▷ Bias-correction of estimates
        $\hat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$
        $\hat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$
        $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$        ▷ Update parameters
    end while
    return $\theta_t$        ▷ Optimized parameters are returned
end procedure
```

MSE / L2 Loss / Quadratic Loss	$\frac{\sum_{i=1}^{N} \left(y_i - \hat{y}_i\right)^2}{N}$
(Binary) Cross Entropy (average reduction on higher dimensions)	$\frac{\sum_{i=1}^{N} \sum_{j=1}^{C} \hat{y}_i \log\left(y_{i,j}\right)}{N}$
Categorical Cross Entropy (sum reduction on higher dimensions)	$-\sum_{i=1}^{N} \hat{y}_i + \log\left(\sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j}\right)$

Table 1.2: y is the output of the network, N is the batch size multiplied by the number of outputs (e.g. pixels), C is the number of classes and $\hat{y}$ is the correct output.

Chapter 2

Fratellì

Hello!

Appendix A

Galileo

```python
import os
os.system("echo 1")
```

$\mathcal{O}\left(n \log n\right)$

`numpy`

Bibliography

[1] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok. «A Novel Ultrathin Elevated Channel Low-temperature Poly-Si TFT». In: 20 (Nov. 1999), pp. 569–571 (cit. on p. 1).