Optimising Gunshot Detection in Tropical Forests for Wildlife Protection with TinyML

ALEXANDRE BISMUTH
École Polytechnique
St Anne's College, University of Oxford

Supervised by Professor Alex Rogers

April 1, 2025

Motivation

- Unsustainable hunting in tropical forests accelerates biodiversity loss
- Machine learning can help enforce wildlife protection laws via gunshot detection.
  1. Field recordings to locate hunting spots at low-costs.
  2. Embedded devices for real-time protection with radio signalling.
- Current systems are not viable for practical applications
  1. Datasets with unrealistic gunshots and a lack of diverse background sounds.
  2. Absence of light models for embedded devices.

Methodology

Dataset - Timestamped gunshot and background sounds from tropical forest in Belize

- Weighted sampler for class imbalance

Metrics - Balance of false positives (costly interventions) and false negatives (risk-rewards incentives) + robustness to class imbalance

- *F1 Score* - Harmonic mean of precision and recall
- *AUPRC* - Area under plot of precision against recall across all decision thresholds

Project Contributions

1. **Analytical framework**: Crafting a ReLU network that approximates a gunshot signal with any error, providing indications regarding minimal model complexity.

2. **Unconstrained pipeline**: Optimising gunshot detection through experimentations with various preprocessing method and adjustments in model architectures.

3. **Lightweight architecture**: Designing gunshot detection systems for embedded devices by combining efficient design choices with model compression methods.

Background

TinyML and Model Compression

Large models are unsuited for edge devices and low-cost machine learning

TinyML - Lightweight models for low-power inference for prolonged monitoring

- Supported by recent advances in hardware and software

Model compression - Retaining high performances while reducing model size

- *Quantization* - Reducing weights and activation values precision
- *Pruning* - Removing weights, neurons, or filters to speed up inference
- *Knowledge Distillation* - Training a small model to mimic a stronger one

Audio Feature Extraction and Data Augmentation

Audio classifications enable multiple types of preprocessing methods

1. Raw waveforms as a single-channel time series
2. Spectrograms-based methods

Audio Feature Extraction and Data Augmentation

Audio classifications enable multiple types of preprocessing methods

1. Raw waveforms as a single-channel time series
2. Spectrograms-based methods

Data augmentation compensates for lack of diversity and robustness of datasets.

- Use of *time and frequency masks*
- Addition of *Gaussian noise*
- Application of *random affine transforms*

Expressive Power of Neural Networks

Universal Approximation Theorem: Any continuous function on a compact domain $f : K \to \mathbb{R}$ can be approximated arbitrarily well by a neural network.

⚠ This does not provide information about network architecture or how to find it

Study of expressiveness yields **width and depth bounds** on neural networks. It guides model design through **complexity analyses** and **justifies empirical performance**.

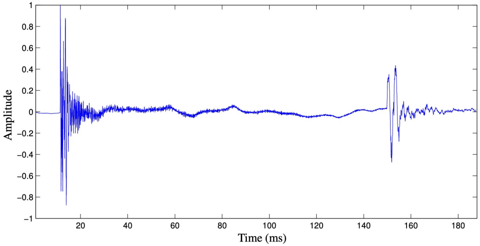
$\Rightarrow$ Proved the potential of ReLU activations

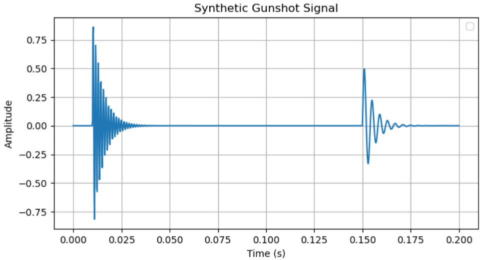
Approximation of a Gunshot Signal using ReLU Networks

Analysis of a Gunshot Signal

The sound of a gunshot is divided in **shock wave** and **muzzle blast**.

- SW: Dispersion of air caused by the bullet *(supersonic)*
- MB: Ignition of gases in the barrel *(sonic)*

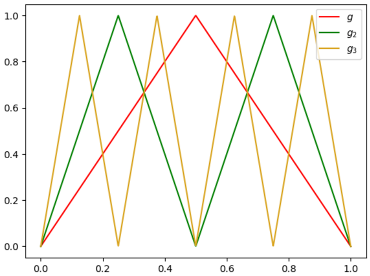


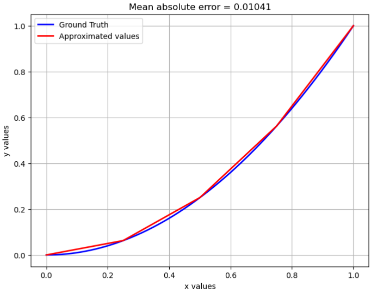


$$f : [0,1] \to \mathbb{R}, \quad x \mapsto e^{-200x} \sin(1500\pi(x - 0.01)) + 0.6 \times e^{-200x} \sin(500\pi(x - 0.15))$$

Setting up elementary functions

Tooth function: $g(x) = 2 \times \mathrm{ReLU}(x) - 4 \times \mathrm{ReLU}(x - 1/2) + 2 \times \mathrm{ReLU}(x - 1)$

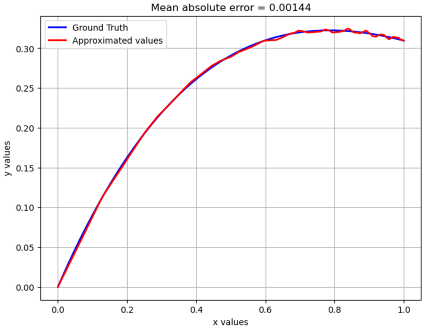




From this, we define **multiplication** as $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$

Approximating $e^{-x}\sin(x)$

Using Maclaurin series with $O(x^{10})$ for e^{-x} and $\sin(x)$, we can approximate $e^{-x}\sin(x)$



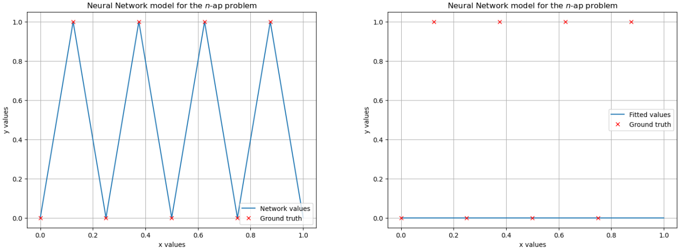
1. Using multiplication, we get $x \mapsto e^{-ax}\ \forall\, a \in \mathbb{R}$.
2. Using Taylor series and ReLU operations, we get $x \mapsto \sin(bx + c)\ \forall\, b, c \in \mathbb{R}$.

Takeaways and Limitations

Network of 864 neurons to approximate a gunshot with $\approx 0.5\%$ error.

- A fairly simple architecture should be sufficient for gunshot detection

Proof of network existence $\not\Rightarrow$ certitude of convergence



⚠ Extreme depth is powerful in theory but not in practice.

- 288 layers yield bad performance due to **gradient vanishing**.

Optimising a Gunshot Detection Pipeline

Updating the Existing Pipeline

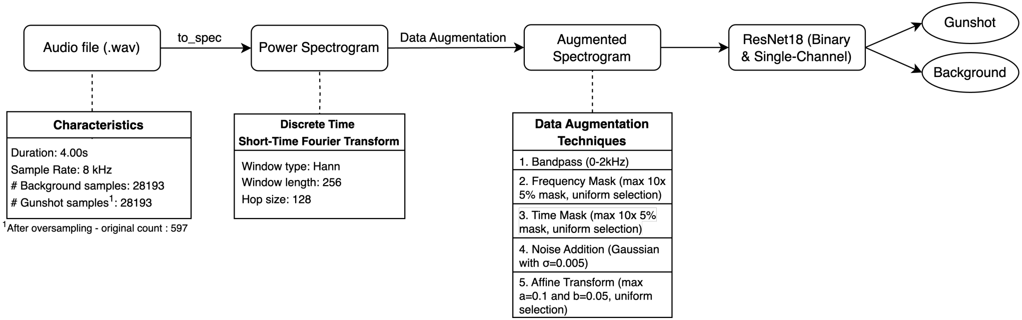
Unconstrained model is essential to study vast areas at low-cost and perform distillation.

- Versioning issues render the old pipeline **unusable** for optimisation and distillation.
- Lacks comparative studies of **preprocessing methods** and **model architectures**.

Updating the Existing Pipeline

Unconstrained model is essential to study vast areas at low-cost and perform distillation.

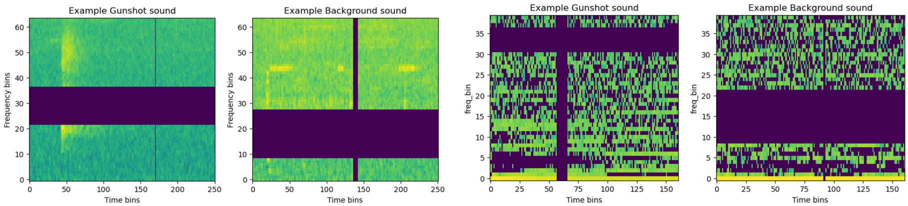
- Versioning issues render the old pipeline **unusable** for optimisation and distillation.
- Lacks comparative studies of **preprocessing methods** and **model architectures**.



F1 score of 0.895 and an AUPRC of 0.934 - *Difference of 10^{-3} with original*

Comparing Preprocessing Methods

1. **Power Spectrograms**: Control ResNet18 - F1: 0.852, AUPRC: 0.884.
2. **Mel Spectrograms**: Ear perception scale - F1: 0.869, AUPRC: 0.890.
3. **Raw Waveforms**: Time-series deep CNN - F1: 0.834, AUPRC: 0.894.
    - Detects sudden jumps but lacks nuance
4. **LFCCs**: Cepstral features, linear filter bank - F1: 0.842, AUPRC: 0.903.
    - Captures spectral energy distribution in a compact yet informative way
5. **MFCCs**: Cepstral features, mel filter bank - F1: 0.840, AUPRC: 0.890.



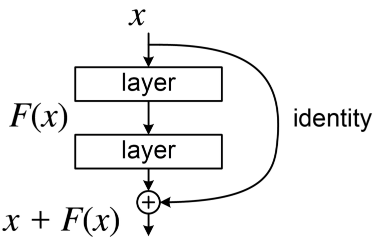
Evaluating Model Architectures and Design Adjustments

ResNets tackle gradient vanishing with **identity connections.**

- Shallower ResNets lack potential
- Deep ResNets lack (qualitative) data

Kernel size and **ratios** can impact perception.

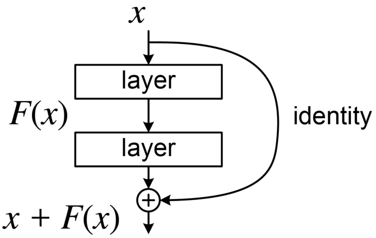
- Larger kernels detect features faster
- Smaller kernels are more precise
- Adjusted y-dim gives increased time attention, helping the model capture sudden changes



Evaluating Model Architectures and Design Adjustments

ResNets tackle gradient vanishing with **identity connections.**

- Shallower ResNets lack potential
- Deep ResNets lack (qualitative) data



Kernel size and **ratios** can impact perception.

- Larger kernels detect features faster
- Smaller kernels are more precise
- Adjusted y-dim gives increased time attention, helping the model capture sudden changes

EfficientNets also improve scaling through **constant ratios.**

- From base model and coefficient ϕ, depth $= 1.1^{\phi}$, width $= 1.2^{\phi}$, resolution $= 1.15^{\phi}$

Evaluating Model Architectures and Design Adjustments

Optimised ResNet

- Kernel size: (3×3)
- Kernel dimensions: (3×15)
- Depth: 50

Optimised EfficientNet

- Kernel size: (3×3) *Default*
- Kernel dimensions: (3×3)
- Scaling coefficient : 3

Model Architecture	**Design Adjustment**	**F1 Score**	**AUPRC**
ResNet18	Default	0.839	0.884
ResNet18	3×3 Initial Kernel	0.863	0.880
ResNet18	3×15 Initial Kernel	0.866	0.880
ResNet50	Default	0.878	0.897
EfficientNetB3	Default	0.887	0.943

Viability of our System - Time Analysis

Time analysis can verify the **viability** of our system

- Consecutive FPs - Some sounds (woodpecker, thunder) are confused with gunshots.
- Consecutive FNs - Rifles categories or gunshots in particular weather go undetected.

Type of Error	**Mean Time Delta**	**Median Time Delta**
False Positives	79 days, 13 hours, 30 minutes	3 hours, 19 minutes
False Negatives	46 days, 2 hours, 27 minutes	30 minutes

Both concerns should be alleviated

Designing a TinyML Gunshot Detection System

Designing a TinyML model

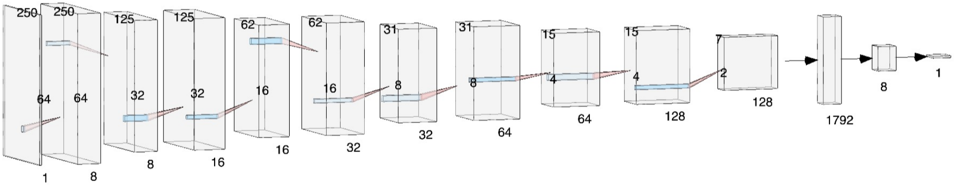
Aim: Design a model optimising the computational resources ($\approx$ 900 kB of storage and 256 kB of RAM) after compression.

- Optimal architecture: $\approx$ 115 thousand parameters

Designing a TinyML model

Aim: Design a model optimising the computational resources ($\approx$ 900 kB of storage and 256 kB of RAM) after compression.

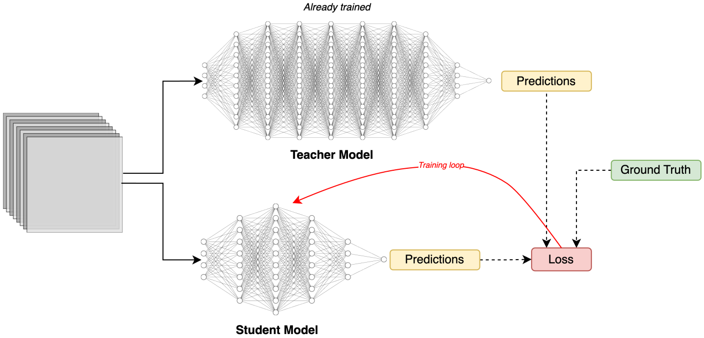
- Optimal architecture: $\approx$ 115 thousand parameters
- Proposed model: Five convolutional blocks with batch normalization and dropout followed by a dense network with sigmoid output



Distilling the Knowledge of a Teacher Model

Distillation improves models by **penalising** differences from a *teacher*'s prediction.

- The student mimics the teacher, which is easier than learning true labels.

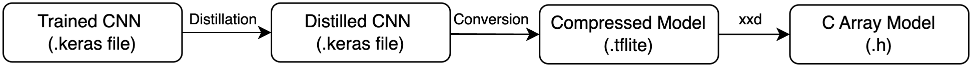


Here, this improves F1 score by 12.0% (from 0.750) and AUPRC by 5.6% (from 0.859).

Deploying to Arduino

Conversion: *Tensorflow Lite* and *C array* with `int8` weights and `float32` activations.

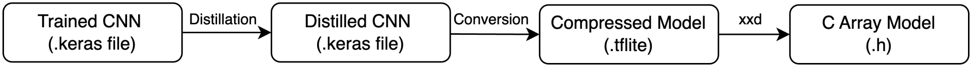
- **Compression efficiency:** 91.4% compression in .tflite and 46.1% compression in .h
- **Resource usage:** 782 kB of storage and 250 kB of peak RAM demand



Deploying to Arduino

Conversion: *Tensorflow Lite* and *C array* with `int8` weights and `float32` activations.

- **Compression efficiency:** 91.4% compression in .tflite and 46.1% compression in .h
- **Resource usage:** 782 kB of storage and 250 kB of peak RAM demand



Inference: Running our model locally with *Arduino TFLite*

- Detects background sounds almost continuously
- Recognises many (but not all) gunshots sounds

Evaluation using an external speaker with ambient noise.

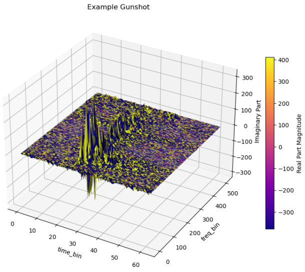
Discussion

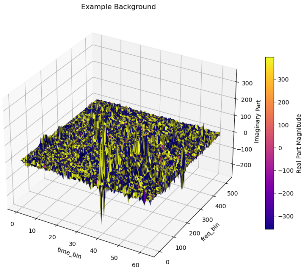
Discussion

Limitation: Our system is only applicable to tropical forests.

This work opens up multiple paths for further research:

1. Creation of a **size bound** for a gunshot network.
2. Test of **alternative devices** for TinyML to optimise cost-performance balance.
3. Study of **novel architectures** like complex neural networks for raw spectrograms.





Conclusion

In summary, we:

1. Approximated a gunshot signal with ReLU networks for any error
2. Optimised an unconstrained gunshot detection pipeline
3. Proposed a TinyML gunshot detection system for Arduino.

These results lead us to the following conclusions:

- Preliminary studies using audio loggers can identify high-risk areas at low-costs
- A TinyML system for wildlife protection in tropical forests should be viable.
- Partnerships with local authorities would help enforce biodiversity protection laws.
  - $\Rightarrow$ A 5% to 10% improvement is still necessary to manufacture a credible system.

Conclusion

In summary, we:

1. Approximated a gunshot signal with ReLU networks for any error
2. Optimised an unconstrained gunshot detection pipeline
3. Proposed a TinyML gunshot detection system for Arduino.

These results lead us to the following conclusions:

- Preliminary studies using audio loggers can identify high-risk areas at low-costs
- A TinyML system for wildlife protection in tropical forests should be viable.
- Partnerships with local authorities would help enforce biodiversity protection laws.
  - $\Rightarrow$ A 5% to 10% improvement is still necessary to manufacture a credible system.

Thank you for listening!

References on the next slide

References I

[1] Angelo MCR Borzino et al. “Gunshot signal enhancement for DOA estimation andweapon recognition”. In: *2014 22nd European Signal Processing Conference (EUSIPCO)*. IEEE. 2014, pp. 1985–1989.

[2] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. “A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[3] Wei Dai et al. “Very deep convolutional neural networks for raw waveforms”. In: *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2017, pp. 421–425.

[4] Robert David et al. “Tensorflow lite micro: Embedded machine learning for tinyml systems”. In: *Proceedings of Machine Learning and Systems* 3 (2021), pp. 800–811.

[5] Amir Gholami et al. “A survey of quantization methods for efficient neural network inference”. In: *Low-power computer vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[6] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[8] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

References II

[9] Lydia Katsis et al. *Tropical forest gunshot classification training audio dataset*. Mar. 2022. URL: https://eprints.soton.ac.uk/455988/.

[10] Lydia KD Katsis et al. "Automated detection of gunshots in tropical forests using convolutional neural networks". In: *Ecological Indicators* 141 (2022), p. 109128.

[11] Douglas A Lyon. "The discrete fourier transform, part 4: spectral leakage". In: *Journal of object technology* 8.7 (2009).

[12] Alex Morehead et al. "Low cost gunshot detection using deep learning on the raspberry pi". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 3038–3044.

[13] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[14] Chiheb Trabelsi et al. "Deep complex networks". In: *arXiv preprint arXiv:1705.09792* (2017).

[15] Shengyun Wei, Shun Zou, Feifan Liao, et al. "A comparison on data augmentation methods based on deep learning for audio classification". In: *Journal of physics: Conference series*. Vol. 1453. 1. IOP Publishing. 2020, p. 012085.

[16] Dmitry Yarotsky. "Error bounds for approximations with deep ReLU networks". In: *CoRR* abs/1610.01145 (2016). arXiv: 1610.01145. URL: http://arxiv.org/abs/1610.01145.

Appendix A.1 : Computing Spectrograms

For a discrete-time signal $x[n]$ with window $w[n]$ and hop size R, the discrete STFT is:

$$X(m,k) = \sum_{n=-\infty}^{+\infty} x[n]\, w[n-mR]\, e^{-j\frac{2\pi}{N}kn}$$

where:

- m indexes the time frames,
- k indexes the frequency bins,
- R is the hop size (i.e., the shift between successive windows),
- N is the FFT size (or the number of frequency bins).

Appendix A.2 : Explaining Cepstral Coeffcients

MFCCs (and LFCCs) transform from spectrograms via the **Discrete Cosine Transform**:

$$c_m = \sum_{k=0}^{K-1} \log(S_k) \cos\left[\frac{\pi m}{K}\left(k + \frac{1}{2}\right)\right], \quad m = 0, 1, \ldots, M-1$$

Where:

- c_m is the m-th MFCC coefficient and S_k is the k-th mel-frequency bin.
- K is the number of Mel bins (or the size of the input vector for DCT).
- M is the number of desired MFCCs (often M is smaller than K).

MFCCs capture the **spectral envelope** and can be interpreted as the spectrum of a log-amplitude spectrum

Appendix B : Function Approximation and Sobolev Spaces

We consider the **Sobolev space**

$$\mathcal{W}^{1,\infty}([0,1]) = \left\{ f : [0,1] \to \mathbb{R} \;\middle|\; f \text{ is weakly differentiable, } \|f\|_{L^\infty([0,1])} < \infty, \text{ and } \|f'\|_{L^\infty([0,1])} < \infty \right\}.$$

We also define the unit ball

$$F_{1,1} = \{f \in \mathcal{W}^{1,\infty}([0,1]) : ||f||_{\mathcal{W}^{1,\infty}([0,1])} \leq 1\}$$

with $\|f\|_{W^{1,\infty}([0,1])} = \max\left\{\operatorname{ess\,sup}_{x\in[0,1]} |f(x)|,\ \operatorname{ess\,sup}_{x\in[0,1]} |f'(x)|\right\}$

Theorem : $\exists$ depth-6 ReLU network with $\frac{c}{\varepsilon \ln(1/\varepsilon)}$ weights that can approximate functions $f \in F_{1,1}$ of a Sobolev space with error ε.

Applications : TinyML architecture and Safety-critical use-cases.

Appendix C : Alternative Model Architectures

Transformer Models: Self-attention mechanisms to weigh different parts of the input.

- **Locality:** CNNs better capture local variations thanks to spectrogram time bins.
- **Data:** Transformers require large datasets to train effectively. For audio detection with limited data, this leads to overfitting and inefficient training.
- **Model Complexity:** Transformers' quadratic scaling is prohibitive to TinyML.

Support Vector Machines: Optimal hyperplane to separate classes in a feature space with kernels which project data into higher dimensions for better class separation.

- **Scalability:** Mel Spectrograms yield high-dimensions features unsuited to SVMs.
- **Hierarchy:** SVMs do not learn hierarchies, limiting deep pattern recognition.
- **Adaptability:** SVMs require careful kernel selection and tuning.

Appendix D : Quantized Distillation

Quantized Distillation : Quantization memory savings + distillation performance.

Algorithm 1 Quantized Distillation

1: Let w be the network weights
2: **loop**
3: $\quad w^q \leftarrow \text{quant\_function}(w, s)$
4: $\quad$ Run forward pass and compute distillation loss $l(w^q)$
5: $\quad$ Run backward pass and compute $\frac{\partial l(w^q)}{\partial w^q}$
6: $\quad$ Update original weights using SGD **in full precision** $w = w - \nu \cdot \frac{\partial l(w^q)}{\partial w^q}$
7: Quantize the weights and return: $w^q \leftarrow \text{quant\_function}(w, s)$

⚠ No signifiant improvement here in comparison to traditional methods.

Appendix E : Degradation

Definition: Adding more layers $\implies$ worsened training

- **Vanishing/Exploding Gradients:** Gradient "signal" can vanish or explode during backpropagation, making learning inefficient or unstable.
- **Complexity:** Overly deep networks overfit by capturing noise instead of patterns.
- **Data Quantity/Quality:** Has to be in good balance with network complexity to prevent overfitting and training incapacities.

Solutions: Simpler architecture and clever adjustments (e.g. skip connections) are crucial to mitigate these issues.

Appendix F : Performance Metrics

- **F1 Score** - Harmonic mean of precision[1] and recall[2] : $F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
- **AUPRC** - Area under the plot of precision against recall across all thresholds τ:

$$\text{AUC}_{\text{Precision-Recall}} = \int_0^1 \text{Precision}(\text{Recall})\, d\text{Recall}$$

[1]Precision - Ratio of predicted positives that were correct : $\frac{\text{TP}}{\text{TP} + \text{FP}}$

[2]Recall - Ratio of true positives that were identified : $\frac{\text{TP}}{\text{TP} + \text{FN}}$